
Mobile Service Provision in Harsh Environments

Bereitstellung mobiler Dienste in rauen Umgebungen

Zur Erlangung des akademischen Grades Doctor rerum naturalium (Dr. rer. nat.)

genehmigte Dissertation von Kamill Panitzek, M. Sc. aus Strzelce (Polen)

Juli 2014 — Darmstadt — D 17



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik



Telecooperation Lab

Mobile Service Provision in Harsh Environments

Bereitstellung mobiler Dienste in rauen Umgebungen

Dissertation

Zur Erlangung des akademischen Grades Doctor rerum naturalium (Dr. rer. nat.)

Eingereicht von

Kamill Panitzek, M. Sc. aus Strzelce (Polen)

Genehmigt vom

Fachbereich Informatik der Technischen Universität Darmstadt

1. Gutachten: Prof. Dr. Max Mühlhäuser

2. Gutachten: Prof. Dr. Jussi Kangasharju

Tag der Einreichung: 14.07.2014

Tag der Disputation: 03.09.2014

Darmstadt 2014

D 17

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-41580

URL: <http://tuprints.ulb.tu-darmstadt.de/4158>

Dieses Dokument wird bereitgestellt von tuprints,

E-Publishing-Service der TU Darmstadt

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de



Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung – Keine kommerzielle Nutzung – Keine Bearbeitung 2.0 Deutschland

<http://creativecommons.org/licenses/by-nc-nd/2.0/de/>



*Jestem leniwy.
Ale to leniwi ludzie wymyślili koło i rower,
gdyż nie chcieli chodzić bądź nosić różnych rzeczy.
– Lech Wałęsa.*





Erklärung zur Dissertation

Hiermit versichere ich, die vorliegende Dissertation ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 14. Juli 2014

(Kamill Panitzek)



Abstract

Envision the future disaster recovery worker relying on mobile devices to work in a disaster setting. This recovery worker requires and benefits from a great number of provided services, for instance, to navigate, communicate, and map the environment. However, he faces adverse networking and computing conditions, characterized by the obstructed or destroyed geographical region, the overloaded or destroyed communication infrastructure, and the density and movement of involved recovery workers. We call such surroundings *harsh environments*.

The present thesis strives to enable recovery workers to consume versatile (cloud) services on top of a resilient communication infrastructure. A service is a software component, i.e., an application, that provides a distinct functionality to accessing users. Thereby, three key challenges have to be faced that also define the three fields of research: (1) improving the resilience of the communication infrastructure, (2) designing a system capable of service provision, and (3) conceiving detailed simulation models and tools to investigate such systems in harsh environments. In each field of research, we contribute and evaluate novel approaches that work together to achieve the main goal of this thesis: *mobile service provision in harsh environments*.

First, we contribute *CityMesh*, a resilient emergency communication infrastructure created by interconnecting wireless routers from private households and public institutions found in urban areas. The resulting decentralized mesh network is based on the IEEE 802.11 WLAN standard and outperforms the existing TETRA standard in terms of data throughput, openness, and failure resilience. Furthermore, we present our research methodology to evaluate the feasibility of our concept by collecting data about wireless routers, enhancing the data accuracy using multilateration, and constructing and analyzing CityMesh networks. We show that network topologies in five selected cities rely on router locations specific to the respective city. Furthermore, we demonstrate that private routers are crucial to create a resilient CityMesh network. To realize the CityMesh concept, we propose to install an emergency switch into wireless routers. This switch could be triggered by eligible authorities in case of a disaster.

Second, we present the core contribution of our thesis: the *peer-to-peer service overlay* concept that fulfills the requirements of harsh environments. To ensure high quality service consumption, the most crucial system components are service discovery (enables service consumers to access a requested service) and service placement (relocates a service in the network topology to increase service quality). Based on simulations, we demonstrate that our conceived *hop count placement* approach outperforms the state of the art placement approach in terms of consistent service quality for all service consumers. Furthermore, we show that latency is not a suitable quality metric for service placement. We also demonstrate that three state of the art service discovery approaches designed for mobile ad-hoc networks are not capable of handling service mobility introduced by service placement. Moreover, a naive request flooding approach performs best in terms of discovery performance and data traffic efficiency when exposed to peer mobility and frequent service migrations. Finally, we

present our real-world experiment setup *PeerMoS* and contribute four concepts for smooth and efficient service migration.

Third, we contribute our integrated simulation framework *DisVis* to investigate the effectiveness of communication systems in realistic harsh environments. Especially in disaster recovery scenarios, the movement of recovery workers depends on the terrain, the current disaster situation, the mission strategy of respective recovery workers, and the orders received from the command centers. Hence, *DisVis* builds upon an OpenStreetMap-based world model and a fine-grained agent-based mobility and communication model. We adjust the rule sets and parameters of our models with results from our survey targeted at experts from first response departments. As opposed to other state of the art mobility simulators, *DisVis* provides an online interface to connect our framework with state of the art network simulators, such as *PeerfactSim.KOM*.

Zusammenfassung

Der Ersthelfer der Zukunft ist ausgestattet mit mobilen Endgeräten und benötigt zahlreiche Dienste für seinen Einsatz, zum Beispiel um im Katastropheneinsatz zu navigieren, mit anderen Ersthelfern zu kommunizieren und um die Umgebung zu kartographieren. Dabei muss er sich jedoch widrigen Kommunikationsbedingungen stellen. Diese werden anhand einer verwüsteten geographischen Region, einer überlasteten oder zerstörten Kommunikationsinfrastruktur und einer hohen Anzahl beteiligter Ersthelfer bzw. ihrer Bewegungen charakterisiert. Solche Umgebungen werden als *raue Umgebungen* bezeichnet.

Die vorliegende Dissertation strebt danach, Ersthelfern über eine robuste Kommunikationsinfrastruktur den Zugriff auf vielseitige (Cloud-)Dienste zu ermöglichen. Ein Dienst ist dabei eine Software-Komponente (eine Applikation), die den Benutzern eine eindeutige Funktionalität bereitstellt. Dabei müssen drei Herausforderungen angegangen werden, die jeweils einen Forschungsschwerpunkt dieser Arbeit bilden: (1) Erhöhung der Robustheit der Kommunikationsinfrastruktur, (2) Design eines Systems zur Dienstbringung und (3) Entwurf detaillierter Simulationsmodelle und Werkzeuge, um Kommunikationssysteme in rauen Umgebungen zu untersuchen. Zu jedem dieser Forschungsschwerpunkte werden neue Ansätze beigetragen, um einheitlich das übergeordnete Ziel zu erreichen: Die *Bereitstellung mobiler Dienste in rauen Umgebungen*.

Der erste wissenschaftliche Beitrag dieser Dissertation ist das *CityMesh*-Konzept. Es handelt sich dabei um eine robuste Notfall-Kommunikationsinfrastruktur, die durch das Verbinden von Drahtlosroutern in urbanen Räumen erzeugt wird. Das so entstehende dezentrale Mesh-Netzwerk basiert auf dem IEEE 802.11 WLAN Standard und übertrifft den heute gebräuchlichen TETRA-Standard in Bezug auf Datendurchsatz, Offenheit und Fehlertoleranz. Zur Untersuchung solcher CityMesh-Netze wird eine Forschungsmethodik vorgestellt, die aus dem Sammeln von Daten über Drahtlosrouter, der Datenaufbereitung und dem Erstellen und Analysieren von CityMesh-Netzen besteht. Bei der Untersuchung von CityMesh-Netzen in fünf Großstädten wird gezeigt, dass diese Netzwerktopologien von den Charakteristiken der jeweiligen Städte abhängig sind. Außerdem zeigt diese Dissertation auf, dass private Drahtlosrouter von essentieller Bedeutung sind, um robuste CityMesh-Netze zu erzeugen. Schließlich wird in dieser Arbeit vorgeschlagen, einen Notfallschalter in Drahtlosrouter zu integrieren. Im Falle einer Katastrophe würde dieser Schalter von einer berechtigten Behörde aktiviert.

Der zweite wissenschaftliche Beitrag, der gleichzeitig den Kernbeitrag dieser Dissertation darstellt, ist das Konzept des *peer-to-peer service overlay*, das die Anforderungen rauer Umgebungen erfüllt. Um hochqualitative Dienste zu gewährleisten, sind die beiden Systemkomponenten Dienstsuche (ermöglicht Benutzern den Zugriff auf Dienste) und Dienstplatzierung (replatziert Dienste in der Netzwerktopologie, um die Dienstqualität zu steigern) von essentieller Bedeutung. Mit Simulationen wird demonstriert, dass der entworfene Platzierungsalgorithmus *hop count placement* den aktuellen Stand der Forschung in Bezug auf eine konsistente Dienstqualität für alle Benutzer übertrifft. Weiterhin wird gezeigt, dass Latenz keine geeignete Entscheidungsmetrik für die Dienstplatzierung ist. Eine weitere Un-

tersuchung demonstriert, dass moderne Ansätze zur Dienstsuche, die für mobile Ad-hoc-Netzwerke konzipiert wurden, nicht in der Lage sind, die Dienstmobilität zu bewältigen, die durch die Dienstplatzierung verursacht wird. Vielmehr basiert der einzige Ansatz, der unter solchen Bedingungen gute Ergebnisse erbringt, auf dem naiven Fluten von Dienstanfragen. Schließlich wird ein echter Prototyp namens *PeerMoS* vorgestellt und vier Konzepte beigetragen, um eine reibungslose und effiziente Dienstmigration zu erreichen.

Der dritte wissenschaftliche Beitrag dieser Dissertation ist das integrierte Simulationsrahmenwerk namens *DisVis*. Insbesondere in Katastropheneinsätzen hängt die Bewegung der Ersthelfer von verschiedensten Faktoren ab. Dazu gehören das Gelände, die aktuelle Situation, die Strategie der jeweiligen Rettungskräfte und die empfangenen Befehle von der Einsatzzentrale. Daher baut *DisVis* auf einem OpenStreetMap-basierten Weltmodell und einem agentenbasierten Bewegungs- und Kommunikationsmodell auf. Der Regelsatz und die Parameter dieser Modelle werden durch Umfragen mit Rettungskräften feinjustiert. Im Gegensatz zu etablierten Bewegungssimulatoren definiert *DisVis* eine Schnittstelle, um moderne Netzwerksimulatoren (z.B. *PeerfactSim.KOM*) mit dem Rahmenwerk zu verbinden.

Acknowledgments

During the past years, I received continuous support, guidance, and encouragement from many people. Without them, I would not have been able to complete this dissertation and I thank all of them.

In particular, I would like to thank my advisor Prof. Dr. Max Mühlhäuser (Technische Universität Darmstadt, Germany). We have a long history now and I am very thankful he enabled me to do my dissertation. Especially, I would like to thank Max for his supervision, his effort, his support, and his patience during the past years. It was very inspiring working with him. Furthermore, I want to thank Prof. Dr. Jussi Kangasharju (University of Helsinki, Finland) for being my co-referee. During my studies, I took several of his courses and his work and enthusiasm for decentralized systems fascinated and inspired me. I see him as the α of my journey that is my doctorate. Being my co-referee, thus, makes him also the Ω of this journey and I am very thankful for that.

A big thank you to Dr. Dirk Bradler and Prof. Dr. Thorsten Strufe for guiding me during my bachelor's and master's theses, respectively. Furthermore, their support and many fruitful discussions helped also completing my dissertation. Special thanks to Dr. Immanuel Schweizer. We met during our studies almost ten years ago, became friends, and went down a long road. Although we helped each other, it is time to say a big thank you for all his help, support, and guidance during the last years. I am sure there is more to come in the future.

I want to thank all of my colleagues at Telecooperation and P2P Networks. First of all, my thanks go to Michael Stein, who worked with me as a student, became a colleague and a friend. You are a great guy and I am sure you will accomplish great things. Thank you Roman Lissermann for not only sharing your workplace with me but also for your motivation during the last mile of both our theses. I want to thank Mohamadreza Khalilbeigi, Axel Schulz, and Jens Heuschkel for the great times and lots of fun we had in our office. Special thanks go to Elke Halla for doing the magic in the background for all of TK and for being such a nice, caring, and helpful person. Thank you Simon Olberding, Marcus Ständer, Felix Heinrichs, Niloofar Dezfuli, Benjamin Schiller, Florian Volk, Sascha Hauke, Benedikt Schmidt, Florian Müller, Martin Schmitz, Jan Riemann, and all present and former members of TK and P2P Networks. Working with you all was a great pleasure and inspired my work. Furthermore, I want to thank my fellow researchers from the QuaP2P project. Thank you Max Lehn, Christian Groß, Dominik Stingl, Karsten Saller, Elias Weingärtner, Tonio Triebel, and Muhammad Ikram. It was great working with you. I also want to thank Dr. Aleksandra Kovacevic, Dr. Patrick Mukherjee, Dr. Sebastian Kaune, Prof. Dr. David Hausheer for advising our research group. Finally, I want to thank all the people who worked on the SFB MAKI proposal.

This dissertation would not have been possible without the excellent students who made great contributions to my research projects. Thank you Michael Stein, Henriette Röger, and Tobias Bönning for the great work on the peer-to-peer service overlay simulation models. Thank you Pascal Weisenburger and Denis Caruso for your excellent work on the simulation

models of DisVis. Thank you Gero Seipel and Tobias Bönning for your great work on the CityMesh concept and related projects. Thank you Lars Fritsche, Felix Hammacher, and Julian Wulfheide for improving on the PeerMoS prototype. Also a big thank you to all the students working with me in bachelor programing projects, seminars, and other courses.

A huge thank you to my closest friends for standing by my side all these years. I want to thank all of them for the great times, great parties, much laughter, and a lot of fun. Thank you Basti and Nadine for being my best friends for almost 20 years. Thanks to Alex L., Alex U., Alina, Armin, Flo, Johnny, Marina, Mo, Nils, Oli, Stephan, Stoppi, and Wolf. During ten years at the university I met many people and made new friends. In particular, I want to thank Chris, Alex, Laura, Betim, Bastian, Sebastian, Max, Robert, and Ahmed. Furthermore, thanks to the guys from the physics department for sharing the pain of working at a dissertation: Benni, Tobi, Simon, and Thomas.

Last but not least, I want to thank my entire family. A special thank you to my parents Elzbieta and Christoph. All I have, all I am today is only because of you. This dissertation is nothing compared to what you have accomplished and I am proud of being your son. I thank my sister Martyna and my brother Paul for being the best siblings I could ask for. I love you guys! And the biggest thank you of all to my beloved Alexandra. You are the best thing that ever happened to me and I am so happy and grateful to have you in my life. Thanks for all your love and support. Thanks for believing in me all this time and even in the darkest hours. Thanks for being my soulmate. Oleńka, ja kocham cię.

Contents

List of Figures	xvi
List of Tables	xvii
List of Algorithms	xix
1 Introduction	1
1.1 Scientific Contributions	2
1.2 Thesis Structure	4
2 System Overview	5
2.1 Harsh Environments	5
2.2 Service Provision	8
2.2.1 Requirements for Service Provision	9
2.3 Coarse System Architecture	13
2.4 Formal System Model	16
2.5 Summary	19
3 Related Work	21
3.1 Communication Infrastructure	21
3.1.1 Backbone-based Infrastructure	22
3.1.2 Decentralized Roll-out Infrastructure	23
3.1.3 Pre-deployed Decentralized Infrastructure	25
3.1.4 Results	28
3.1.5 Routing	29
3.2 Distributed Service Provision	31
3.2.1 Service Migration	32
3.2.2 Approaches for Service Provision in Harsh Environments	33
3.2.3 Service Placement	39
3.2.4 Service Placement Evaluation Baseline	41
3.2.5 Service Discovery	43
3.2.6 Selected Service Discovery Candidates	48
3.2.7 Other System Components	51
3.3 Simulation Models	54
3.3.1 Mobility	54
3.3.2 Communication	59
3.3.3 Simulation Frameworks	62
3.4 Summary	65

4	Communication Infrastructure	67
4.1	Privatized Pre-deployed Infrastructure	67
4.1.1	General Concept ‘CityMesh’	68
4.1.2	Emergency Switch for Wireless Routers	69
4.2	Methodology to Investigate the Feasibility of CityMesh Networks	70
4.2.1	Data Collection of Public Routers from Online Sources	71
4.2.2	Data Collection of Private Routers with Wardriving	72
4.2.3	Construction of CityMesh Networks	75
4.2.4	Graph-theoretical Analysis of CityMesh Networks	76
4.3	Feasibility Evaluation of CityMesh Networks	77
4.3.1	Random Network Topologies vs. CityMesh Networks	77
4.3.2	Impact of Private Routers	82
4.3.3	Conclusion	86
4.4	Discussion of Realization Aspects	87
4.4.1	Energy Supply	87
4.4.2	Integration and Activation	88
4.4.3	Security and Isolation of Networks	89
4.4.4	Legal Issues and Societal Acceptance	89
4.5	Summary	90
5	Peer-to-Peer Service Provision	93
5.1	System Model	94
5.1.1	Overview	94
5.1.2	Service Discovery	95
5.1.3	Service Placement	96
5.1.4	Simulation Components	97
5.2	Service Placement	100
5.2.1	Latency-based Service Placement	101
5.2.2	Hop-based Service Placement	103
5.2.3	Evaluation Setup	107
5.2.4	Evaluation of the Single Group Setup	109
5.2.5	Evaluation of the Command Center Setup	111
5.2.6	Conclusion	119
5.3	Service Discovery	120
5.3.1	Evaluation Setup	121
5.3.2	Evaluation of Discovery Performance with Static Service Placement	122
5.3.3	Evaluation of Discovery Performance with Service Mobility	125
5.3.4	Conclusion	129
5.4	Smooth Service Migration	130
5.4.1	Real Experiment Setup	130
5.4.2	Graphical User Interface	135
5.4.3	Concepts for Smooth Service Migration	138
5.5	Summary	143

6	Simulation Models and Tools for Harsh Environments	145
6.1	Simulation Models	146
6.1.1	World Model	147
6.1.2	Mobility Model: Movement	147
6.1.3	Mobility Model: Communication and Interaction	149
6.2	Fine Adjustment of Simulation Models	151
6.3	Integrated Simulation Framework	153
6.3.1	Disaster Simulation Framework: DisVis	154
6.3.2	Interlink to Network Simulator	156
6.4	Summary	158
7	Conclusion	159
7.1	Summary of Contributions	159
7.1.1	Communication Infrastructure	159
7.1.2	Peer-to-peer Service Provision	160
7.1.3	Simulation Models and Tools for Harsh Environments	161
7.2	Potential Directions for Future Research	162
	Bibliography	165
	Curriculum Vitae	183



List of Figures

1	Overview of the contributions	3
2	General system overview	6
3	Roles of participants in a service overlay network	11
4	Coarse system architecture of a service overlay	13
5	System design of a service overlay on one peer	19
6	Overview of communication infrastructure concepts for harsh environments .	22
7	Coverage Map of Google WiFi in Mountain View, CA (Screenshot)	26
8	Example network topology for the tree placement approach	42
9	Classification of service discovery approaches by Ververidis et al. [193]	44
10	Directory nodes ('1', '2', and '3') forming the backbone of directories	48
11	Selection of the service brokers ('1', '2', '3', and '4')	49
12	Rendezvous of advertisements (starting at 'P') and queries (starting at 'C') . .	51
13	Bipartite matching example	52
14	General concept of 'CityMesh'	68
15	Methodology to evaluate the feasibility of CityMesh networks	70
16	Area of the wardriving measurement	73
17	Multilateration	74
18	Comparison of the WiGLE data and our wardriving data in Darmstadt	75
19	Example of a unit disk graph with distance unit r	76
20	CityMesh networks of public wireless routers in big cities	78
21	Degree distribution of Melbourne and Mountain View	80
22	Maximum cluster sizes in Melbourne and Mountain View when removing hubs	81
23	Constructed mesh networks in Darmstadt	83
24	Degree distribution for CityMesh networks in Darmstadt	84
25	Number of clusters when removing hubs in Darmstadt CityMesh networks . .	85
26	Maximum cluster sizes when removing hubs in Darmstadt CityMesh networks	85
27	Shortest path length distribution in Darmstadt CityMesh networks	86
28	Extended system model of the service overlay, including simulation components	94
29	UML class diagram of the service discovery components	95
30	UML class diagram of the service placement components	96
31	UML class diagram of service demand generators	98
32	UML class diagram of the service traffic generators	99
33	Placement comparison: tree placement ('T') and latency placement ('L')	101
34	Average hop count for single group setup	109
35	Average mouth-to-ear delay for single group setup using AODV routing	110
36	Message loss rates for single group setup using AODV routing	110

37	Number of migrations for single group setup using AODV routing	110
38	Average hop count for command center setup using Dijkstra routing	112
39	Average hop count for command center setup using AODV routing	112
40	Average mouth-to-ear delay for command center setup using AODV routing .	113
41	Hop count CDFs with 6 group members for command center setup using AODV routing	114
42	Mouth-to-ear delay CDFs with 6 group members for command center setup using AODV routing	114
43	Hop count CDFs per placement approach with 6 group members for command center setup using AODV routing	115
44	Mouth-to-ear delay CDFs per placement approach with 6 group members for command center setup using AODV routing	116
45	Average number of migrations for command center setup using AODV routing	118
46	Average message loss rates for command center setup using AODV routing . .	118
47	Message loss rates for command center setup using AODV routing	119
48	Data traffic on the transport layer generated by different discovery ap- proaches with static service placement and 50 peers	123
49	Data traffic on the transport layer generated by different discovery ap- proaches with static service placement and 150 peers	124
50	Data traffic on the transport layer generated by different discovery ap- proaches with static service placement and 250 peers	124
51	Performance of different discovery approaches with static service placement and a varying number of peers	126
52	Performance of discovery approaches with different service placement strate- gies and 150 peers	128
53	Measurements of 1-hop ad-hoc communication in Kbyte/s	133
54	Measurements of 2-hop ad-hoc communication in Kbyte/s	134
55	PeerMoS – graphical user interface	137
56	Sequence diagram of our smooth service migration concept	139
57	Binary network protocol of PeerMoS in an IPv4 network	142
58	Required simulation models for communications research in harsh environ- ments	146
59	Agent types of our mobility models for disaster recovery	148
60	Rule set of paramedics	150
61	An example message flow in the communication hierarchy of first responders	151
62	Different world models for disaster simulations	154
63	The graphical representation of simulation results	156
64	Simulation setups for communications research	157

List of Tables

1	Comparison of presented infrastructure support approaches for disaster recovery	28
2	Comparison of presented routing protocols	31
3	Comparison of presented approaches for service provision in harsh environments	38
4	Comparison of presented service discovery approaches	47
5	Applicability of presented mobility models to participants in disaster relief scenarios	58
6	Evaluation of presented simulation frameworks	65
7	Properties of all analyzed cities	79
8	Average node degree values for evaluated cities	79
9	Characteristic shortest path length (CPL) of CityMesh networks	81
10	Diameter of CityMesh networks	82
11	Network properties for different communication ranges r	84
12	Network models and parameters	107
13	Serialization measurements of the voice chat service	141
14	Message types of first responder agents	150





List of Algorithms

1	Tree placement by Oikonomou et al. [142]	42
2	Active latency placement	102
3	Passive latency placement	103
4	Hop count placement	105
5	Max hop placement	106



1 Introduction

We envision disaster recovery workers in the future to rely on mobile devices to communicate with their colleagues, to navigate through the disaster area, and to map the environment. These recovery workers require and benefit from a huge variety of provided services. However, they face adverse networking and computing conditions during their missions. The geographical region they work in comprises obstacles (e.g., destroyed buildings and streets) that restrain recovery workers in their movement. Furthermore, the communication system used by recovery workers today has several limitations and drawbacks:

- Access to Internet-based services requires additional hardware and communication infrastructure, which is either destroyed or most commonly overloaded after a disaster [97].
- The communication system does not scale well with the number of first responders involved in the recovery mission as have shown the 9/11 attacks on the Pentagon [56].
- Group communication for small groups of first responders is restricted due to the limited number of communication channels.

To overcome most of these limitations, peer-to-peer-based approaches have been proposed for disaster recovery missions [34, 35, 102, 123]. Most of these novel approaches are based on digital multi-hop wireless communication using WiFi ad-hoc connections. This provides the opportunity to use the resulting mobile ad-hoc network in various ways. For instance, if this ad-hoc network contains a gateway to the Internet, access to Internet-based services is possible. For assistance during recovery missions, first responders could harness cloud services through the network. However, in harsh environments, such as disaster relief, Internet access is not necessarily available. And limited transmission range of wireless devices still remains a problem. This makes access to services in the cloud especially hard.

The present thesis strives to tackle the main challenges to provide versatile services in harsh environments and particularly in disaster recovery:

1. To improve the unreliable communication infrastructure and to overcome the limited transmission range of wireless devices, we propose a city-wide emergency network: the *CityMesh*.
2. To realize service provision without the need for an always online Internet connection, we propose an architecture to reliably provide versatile (software) services in harsh environments: the *peer-to-peer service overlay*.
3. To investigate the influence of new communication technology on the effectiveness of recovery workers, we propose an integrated simulation framework based on a detailed world model and with support for agent-based mobility models: *DisVis*.

The CityMesh network builds a supportive communication infrastructure that could be used during disaster situations. To realize this, we propose an emergency switch to be implemented in public and private wireless routers. Triggered by an eligible authority, this emergency switch transitions the routers into an emergency mode, thus, creating a city-wide wireless mesh network. Especially in urban areas, this CityMesh network would assist first responder communication. At the same time, it would close the communication gap between the command center and the rescue workers on site. We collect data on potential CityMesh networks and analyze them theoretically. Furthermore, we discuss several aspects related to the realization of an emergency switch for wireless routers. Finally, we integrate our findings into our simulation models and use CityMesh networks in our investigations.

Our proposed peer-to-peer service overlay brings the cloud (service) to the service consumers by harnessing peer-to-peer concepts for service provision. We identify the key components of peer-to-peer service overlays and describe a generic system architecture of such a system. Furthermore, we highlight the two most important components for service consumption: service placement and service discovery. Service discovery enables users to access services in the first place. The service placement component then ensures high service quality by dynamically relocating the service instance in the network topology. We investigate both components under realistic conditions using simulations. Finally, we investigate smooth service migrations and demonstrate the feasibility of our concept with a real world prototype, called *PeerMoS*.

DisVis, our integrated simulation framework, focuses on realistic simulation-based investigations of communication systems for disaster recovery. The movement of recovery workers relies on the terrain and the current disaster situation, on the mission strategy of respective rescue workers, and on orders received from the command centers. Therefore, we conceive a detailed world model and an agent-based mobility model that further integrates movement and communication components. Finally, we design an online interface for our simulation framework to be connected with state of the art network simulators.

1.1 Scientific Contributions

The contributions of this thesis are positioned along three research fields: the infrastructure layer of harsh environments, the peer-to-peer service overlay, and the simulation models. A schematic overview of the contributions is also depicted in Figure 1. Contributions in this figure are colored according to the respective field of research. We summarize the scientific contributions as follows:

1. We contribute *CityMesh*, a resilient emergency communication infrastructure created by interconnecting private and public wireless routers in urban areas [150, 151, 152, 153]. To realize this emergency communication infrastructure, we propose an emergency switch to be integrated in private and public wireless routers. For the feasibility evaluation of CityMesh networks, we propose a methodology that consists of (1) data collection about wireless routers, (2) data enhancement using multilateration, and (3) construction and analysis of CityMesh networks. We show that networks in five selected cities rely on the geographical and architectural characteristics of the respective city. Furthermore, we demonstrate that only with private wireless routers the density of routers in the city center of Darmstadt is high enough to create a resilient CityMesh network assuming low communication ranges.

2. We contribute our *peer-to-peer service overlay* concept for service provision in harsh environments [147, 154]. It consists of several components necessary for service provision in harsh environments. We highlight the importance of the service placement and the service discovery components. We propose four service placement strategies to provide consistent service quality in terms of mouth-to-ear delay for all service consumers. By investigating them in disaster recovery settings, we demonstrate that our conceived hop count placement approach outperforms the state of the art service placement in terms of consistent service quality. We further demonstrate that three selected state of the art service discovery approaches are not capable of handling service mobility introduced by service placement. Finally, we present our real-world experiment setup *PeerMoS* and contribute four concepts for smooth and efficient service migration [148, 149]: soft handover, persistent socket connections, efficient serialization, and an efficient network protocol.

3. We contribute our integrated simulation framework *DisVis* to investigate the effectiveness of communication systems in realistic harsh environments [155]. It builds upon an OpenStreetMap-based world model and a fine-grained agent-based mobility model. We adjust the rule sets and parameters of our models with results from our survey targeted at experts from first response departments. To enrich the simulation with realistic network models, we define an interface to connect our framework with state of the art network simulators, such as PeerfactSim.KOM.

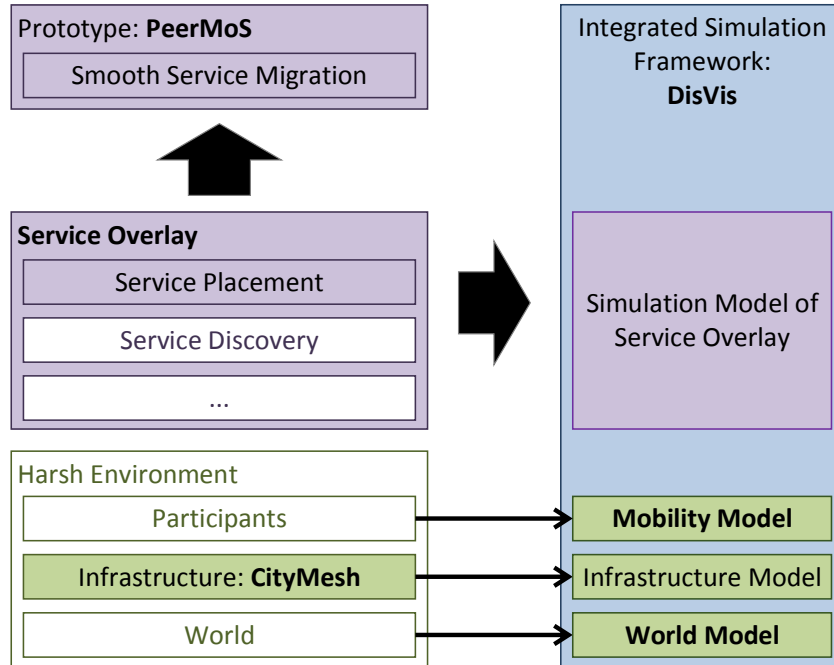


Figure 1: Overview of the contributions

1.2 Thesis Structure

In Chapter 2, we present an overview of the entire system for service provision in harsh environments. This system consists of the harsh environment, the service overlay, and simulation models. We define harsh environments and deduce functional and non-functional requirements for service provision. Finally, we present a coarse system architecture and a formal system model, both describing service overlays. We investigate state of the art concepts and approaches related to this thesis in Chapter 3.

In Chapter 4, we introduce our concept of the CityMesh, a supportive communication infrastructure for first responder communication. We propose an emergency switch to be implemented in public and private wireless routers that enables first responders to use CityMesh networks during disaster recovery missions. We collect data of wireless routers in selected cities and evaluate potential CityMesh networks for their graph-theoretical properties. Finally, we also discuss realization-related aspects of our proposed emergency switch for wireless routers.

In Chapter 5, we describe the peer-to-peer service overlay concept in more detail. We propose four service placement approaches to provide consistent service quality to all connected service consumers and evaluate these approaches against a state of the art placement strategy. Furthermore, we evaluate the impact of service placement on state of the art service discovery candidates. Finally, we present four concepts for smooth service migration and investigate them in a real-world experiment setup, called PeerMoS.

In Chapter 6, we present our approach to simulate mobility and communication in disaster recovery scenarios based on expert knowledge and state of the art work. We present our mobility model that is based on the concept of software agents. It uses a detailed world model that we create from real-world OpenStreetMap data. Furthermore, we discuss the benefits of the integrated network and mobility simulation concept and present our integrated simulation framework.

We summarize this thesis in Chapter 7 and discuss possible future research directions based on the contributions of this thesis.

2 System Overview

The last chapter introduced to the multi-faceted topic of service provision in harsh environments. We call our approach to service provision the *peer-to-peer service overlay* (cf., Chapter 5). In the present chapter, we start with a general overview on the entire system to better differentiate between the service overlay and the harsh environment. We define constraints of harsh environments and present our general approach to service provision. Finally, we also formalize our peer-to-peer service overlay.

In the following sections, we first define harsh environments including all three environmental parts (Section 2.1). Afterward, we define service provision in the context of this thesis and deduce requirements for service provision in harsh environments (Section 2.2). A first coarse system architecture for service provision is presented in Section 2.3. We provide a formal system model of the relevant system parts in Section 2.4. Finally, Section 2.5 gives a short summary on the system overview.

2.1 Harsh Environments

Figure 2 provides an overview on the entire system. It consists of two major parts: the peer-to-peer service overlay and the environment. The service overlay is depicted as the top most layer. It is the distributed software application running on appliances carried by the participants and also running on stationary devices in the infrastructure. Everything except the service overlay is defined to be the environment where the service overlay is deployed and executed in. Finally, simulation models are essential to examine our proposed service overlay in harsh environments (cf., Chapter 6). They cover all layers depicted in Figure 2.

The environment consists of three layers: the geographical region, the static communication infrastructure in that region, and the participants. The participants are interacting with the service overlay through their equipment. They form a mobile ad-hoc communication network and access the service overlay via this network. This ad-hoc network is augmented by means of additional infrastructure components (cf., Chapter 4). The infrastructure is optional but some degree of infrastructure support is assumed throughout this thesis. This assumption is in line with common experiences from large disasters of recent years [25, 59, 140]. The infrastructure can improve the connection quality between participants. It also can increase the reachability of participants that are detached from (parts of) the mobile ad-hoc network. In the following, we describe three environmental parts in context of the example application scenario of disaster recovery:

Geographical region: The geographical region is the surrounding where a given scenario is situated in. In harsh environments, the geographical region comprises obstacles that restrain participants in their movement. In disaster recovery scenarios, destroyed buildings and streets are common obstacles. Furthermore, the region can comprise buildings and radio sources interfering with the wireless communication between participants. Buildings and other objects can reflect radio signals, hence, degrade the

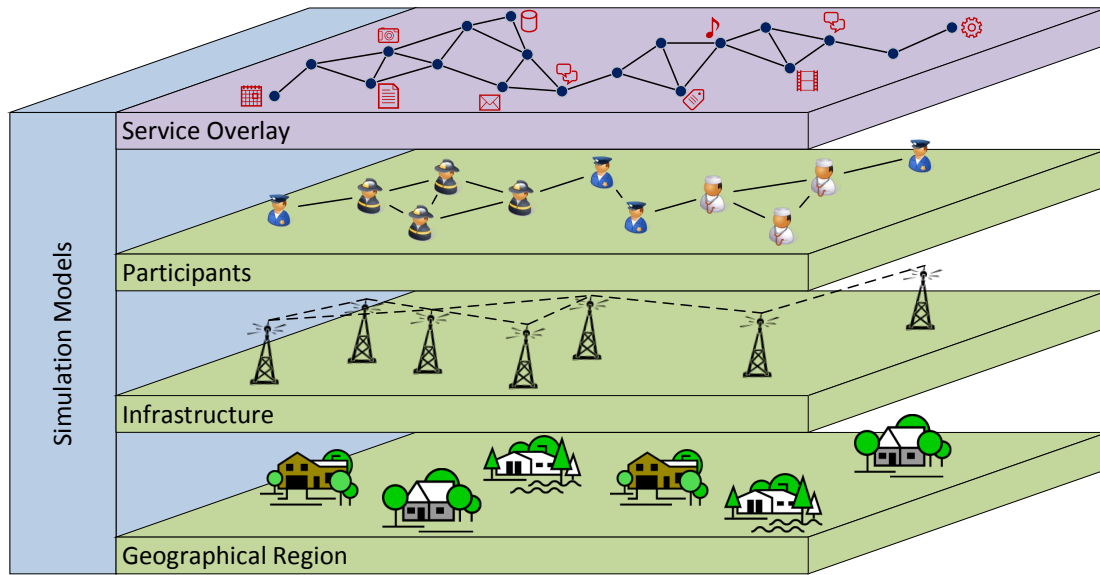


Figure 2: General system overview

quality of wireless communication. Urban areas with high buildings are especially affected by these factors.

Communication Infrastructure: The communication infrastructure consists of several stationary devices deployed in the geographical region. In this thesis, we assume these devices to be WiFi routers. The infrastructure can be used by participants for communication purposes among each other by connecting their devices to the infrastructure. It can also be used to connect to the Internet and to communicate with the outside world. However, in harsh environments and especially in disaster recovery, parts of the infrastructure are highly unreliable due to destruction or communication overload.

Participants: Participants are people interacting with the service overlay and also with the entire environment. This includes all three parts of the environment: the geographical region (e.g., extinguishing fire), the infrastructure (e.g., for connecting to the service overlay), and other participants (e.g., calling for help). Participants carry (mobile) devices which are interconnected through wireless links, thus, creating a local ad-hoc network. This network allows participants to communicate with each other and to consume versatile services. Participants are mobile and move around the area. While moving, wireless links between devices can break if participants move out of the wireless transmission range of other devices. However, high density of participants with very slow movement makes an environment harsh, too. High density of participants and a high communication demand on the shared wireless medium can lead to interferences and communication loss due to overload.

In summary, harsh environments are defined by the properties of the three environmental parts: the density and movement of participants, the overloaded or destroyed communication infrastructure, and the obstructed or destroyed geographical region. As described

above, the communication devices of participants as well as the infrastructure create a mobile ad-hoc network. This network constitutes the basis for communication among participants and for service provision. We call this network the *underlay network*.

Definition 2.1. An **underlay network** is represented by a directed, loop-free graph $\mathcal{G} = (V, E)$ of vertices $V = \{v_1, \dots, v_m\}$ (physical networking hosts and routers) connected by edges $E \subseteq V \times V \setminus \{(v, v) | v \in V\}$ (physical network connections). Hosts u, v with a physical network connection $(u, v) \in E$ are called neighbors. For underlay networks in harsh environments, $\forall (u, v) \in E : (v, u) \in E$ does not necessarily apply, i.e., not all links are bi-directional.

In harsh environments, physical networking hosts (vertices) are a mix of wireless routers and hand held devices. Physical connections (edges) between hosts are wireless links. Therefore, and due to heterogeneous hardware and different transmission powers, hosts have individual communication ranges. A host u might be able to send messages to host v while transmission power of host v might not be sufficient to send messages back to host u . However, signal attenuation is the main reason why links between hosts are not necessarily bi-directional.

Vertices and edges in the underlay are labeled with attributes defining their available resources, limitations, and costs in terms of resource consumption. Tuples of attributes are assigned to hosts to describe available hardware and software resources of hosts such as CPU, memory, storage capacity, energy supply, operating system, etc. A capacity function defines the capacity of a single resource on individual hosts (e.g., computational power of hosts). For example, service provision considers the number of service instances executable at a discrete point in time on a particular host. This is limited by the computational power of the host. Edges are assigned a tuple with parameters such as bandwidth, delay, packet loss probability, etc. An important parameter of communication links is the latency describing how fast both hosts respond to each other's requests.

Definition 2.2. For the set B of **resource capacities** (such as CPU, memory, etc.), the function $\beta : V \rightarrow \mathbb{R}_+^{|B|}$ assigns each underlay host a tuple of its capacity values. For the set W of **connection parameters** (such as bandwidth, delay, etc.), the function $\omega : E \rightarrow \mathbb{R}_+^{|W|}$ assigns each underlay connection a tuple of its parameter values.

Since connected (hand held) devices are carried by participants, the resulting topology of the underlay network is exposed to high dynamics and frequent changes. Hence, the underlay model must consider the mobility of network hosts (participants and their devices), too. In this respect, we simplify the movement and only assume movement on a two-dimensional plane. This plane can be represented by a Cartesian coordinate system and networking hosts are assigned points on this plane dependent on the time. Furthermore, the connections between hosts break, if they move out of each other's wireless transmission range. Therefore, the links between hosts must be time-dependent, as well:

Definition 2.3. For underlay networking hosts V and any point in time $t \in T$ the **positioning function** $\phi : V \times T \rightarrow \mathbb{R}^2$ assigns a position on a two-dimensional plane to the underlay hosts. Also, $E(t)$ is defined as the subset of all active underlay connections $E(t) \subseteq E$ at time t .

Furthermore, we call an environment *harsh*, if the underlay network is negatively affected due to the environmental properties or (frequent) changes of said properties:

Definition 2.4. We define a **harsh environment** as an obstructed or partly destroyed geographical surrounding where existing communication infrastructure is destroyed or overloaded. The density and movement of participants additionally add signal interference to the wireless communication medium and dynamics to the overall system. In harsh environments, hosts (devices of participants) have low resource capacities $b \in B$ and underlay connections have low quality connection parameters $w \in W$.

A prominent example for application scenarios in harsh environments is disaster recovery. (Nature or man-made) disasters are often accompanied by destruction, injuries, and the loss of human lives. Shortly after a disaster has struck, rescue workers arrive at the scene and start searching for survivors. To coordinate themselves, they use wireless communication, if possible [140]. Destruction of communication infrastructure is common, thus, limiting the means for communication. Therefore, rescue workers often have to deploy their own communication infrastructure [25].

Another example for scenarios in harsh environments are mobile-assisted public gatherings. In such gatherings, people visit a concert, a sports match, or any other large scale live event. Although, people at live events do not usually move much, the local communication infrastructure is quickly overloaded due to the high communication demand [175]. During such events visitors want to take pictures, videos, and other recordings to share them with acquaintances at home or at the same or a parallel event. Sharing their media content with others can help visitors with a limited view to follow the event. Also, the local organizer can distribute information about the event to the visitors. Furthermore, small games or location sharing, can be provided to and amongst visitors.

Throughout the remainder of this thesis, we use disaster recovery in urban areas as an example scenario in harsh environments. Because the environmental constraints are tighter in such scenarios and can also cover other application scenarios, e.g., mobile-assisted gatherings.

2.2 Service Provision

We defined the properties of harsh environments: obstructed geographical region, overloaded or destroyed communication infrastructure, and fluctuating density and movement of participants who, at the same time, interact with the service overlay. Next, services and service provision in the context of this thesis have to be defined. Furthermore, functional and non-functional requirements for service provision in harsh environments must be defined. They should be deduced from the definitions above and target the application scenario.

Before deducing the requirements for service provision in harsh environments, we first define services and associated terminology. This also includes the roles of participants and the service overlay. We define services in the context of this thesis as follows:

Definition 2.5. A **service** is a software component, i.e., a program executed on a computing device. This software service provides a distinct functionality to accessing entities. A running software service is called a **service instance**.

Furthermore, we define the different roles of participants and devices in a system for service provision as follows:

Definition 2.6. *The device executing a software service is called **service provider** or **service hoster**. And a participant accessing, consuming, or interacting with a service is called **service consumer**.*

Definition 2.5 also implies a service to provide a client interface stub for a service consumer to interact with the service. The client interface stub is not necessarily integrated into the software service itself. It can be implemented as a separate module that is linked to the respective service. To interact with the service remotely, a service consumer first obtains the respective client interface stub automatically from the system. We call this system a *service overlay*:

Definition 2.7. *The **service overlay** is a distributed software application running on appliances carried by the participants and also running on stationary devices in the infrastructure. It handles queries for services, manages access to requested services, and provides means for adding and removing services from the service overlay.*

Finally, software services can be combined to create powerful applications. For instance, by combining a text chat service with a picture and video sharing service a rich chat application can be created. In the remainder of this thesis, we make heavy use of (voice and text) chat services as example services. Especially the voice chat service is important during rescue missions because rescue workers give orders and coordinate themselves using voice transmission. Other examples for services are file storage services, data exchange services, or gaming services, however, in different application scenarios.

2.2.1 Requirements for Service Provision

In our example scenario, first responders require versatile services during disaster recovery missions. For instance, hierarchical group communication using text, voice, or video chats helps limiting communication to only involved rescue workers. Also, a service for automatic information gathering and sharing helps to organize the mission and to warn rescue workers in dangerous or hazardous situations.

In the past, the requirements for emergency response communication systems have been formulated multiple times [21, 33, 35, 50, 140]. In the following, we define functional requirements followed by the presentation of non-functional requirements specific to service provision in harsh environments. More precisely, the main goal is to help rescue workers during their mission by enabling them to access a variety of services. This makes rescue workers the primary service consumers in this application scenario. Our proposed service overlay is deployed as an application on top of an emergency response communication system, hence, the requirements formulated in previous works still remain valid.

Functional Requirements

Functional requirements describe the prerequisites that enable first responders to access our proposed service overlay and to consume provided services. First of all, rescue workers must be able to access and consume services. Therefore, they need to carry some kind of computing device. Also, an appropriate interface for service consumption is needed at the

service. The easiest way to consume a service is to transfer this service to the own device and provide it locally. Hence, computing devices of rescue workers must be able to execute software services. We assume every rescue worker carries a mobile device through which he can access, consume, and provide services. In a final implementation, such devices might be integrated into the equipment, for instance, a head-mounted display integrated into the helmet of a fire fighter.

Requirement 2.1. Participation: Every potential service consumer carries a device capable of software service provision to participate in the service overlay. This allows service consumers to consume and provide services.

As described above, the environment is highly dynamic. Rescue workers move across the disaster area and use wireless connections to communicate with other team members as well as with the command center. Therefore, they must be able to consume services seamlessly while moving. This requires a network of mobile (and stationary) computing devices. This network is then used by participants of the service overlay to access remote services and to exchange services between participants.

Requirement 2.2. Network: A network of interconnected computing devices is required for participants to access services provided in the service overlay. Service consumers use locally provided services (on their own device) or access remote services via network connections.

The service overlay must be able to handle multiple and diverse services. It also must provide means to add new services into the system as well as to remove obsolete services from the system. Furthermore, a potential service consumer must be able to search for a service matching his needs. This implies that services are searchable, addressable, and accessible in the network. And if they exist, services must be available and consumable upon request.

Requirement 2.3. Access methods: The service overlay must provide means for adding, searching, addressing, and accessing software services.

The different roles of participants and the network formed between them are depicted in Figure 3. In this case, the service consumer and the service provider are two different participants. The service consumer connects to the service provider through the network to consume the provided service.

Individual participants and their devices must be able to provide and, therefore, execute a subset of available services in the network. Since devices are most probably heterogeneous in terms of their capabilities and hardware resources, services might not be executable on every device. For example, a mobile server inside a fire truck might have a more powerful CPU and more memory than the integrated mobile device carried by a fire fighter. Also, the software component providing the actual service is not required to be executed locally on the service consumer's device. Therefore, network connectivity and communication between service consumer and service provider is required during service consumption. This is especially needed if the service requires resources which are not available on the local device and, therefore, must be executed remotely.

Requirement 2.4. Resource matching: Due to heterogeneous capabilities and hardware resources of devices and versatile resource requirements of services, the service overlay must

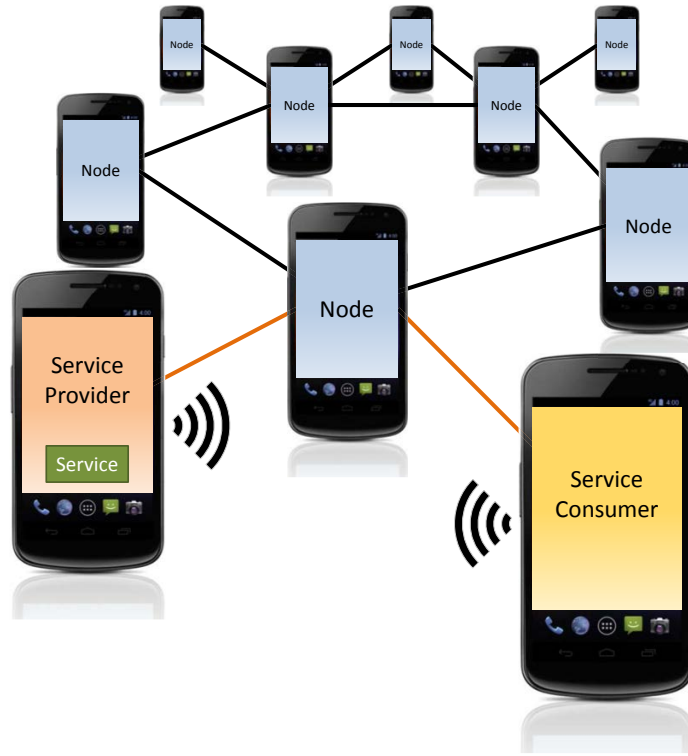


Figure 3: Roles of participants in a service overlay network

automatically choose a suitable service provider for a given service. Service consumers must be able to access remotely provided services through network connections.

Finally, the quality of provided services depends on various factors. The capabilities and hardware resources of the service provider define the quality of provided services. And, if the service is provided remotely, the network connection between service consumer and service provider usually has an influence on the service quality, too. Especially during rescue missions, the quality of service should not degrade under a certain level or human lives can be threatened. The different factors that incorporate into the quality of a service are described in the next section. These factors are the non-functional requirements.

Requirement 2.5. Service quality: Along with the resource requirements of services, services also define metrics to measure their service quality. During service provision, the service overlay must continuously monitor the quality of provided services and trigger respective actions to ensure highest service quality possible.

Non-functional Requirements

After describing the functional requirements of our proposed service overlay, we define the non-functional requirements specific to harsh environments. This section is based on parts from our chapter “Service Overlays” [149] in “Benchmarking Peer-to-Peer Systems” [60]. We

especially make use of the concept of quality aspects to describe the non-functional requirements. Focusing on the main functionality of service overlays (cf., functional requirements above), we consider the quality aspects *performance*, *minimum costs*, *efficiency*, *robustness*, and *scalability*. In the following, we describe these quality aspects in more detail:

Performance. In terms of access methods of the service overlay, performance is specified as how fast and how accurate the service overlay responds to search queries. In terms of service provision, performance is specified as how fast services respond to requests. We also use responsiveness of services as a synonym for performance. Often, clients interact with services in real-time, hence, they require highly responsive services. This involves properties of both the network connection between service consumer and service provider and the computing resources of the service provider. Requests and responses must reach the service provider and the service consumer, respectively, with minimal delay. And requests must be processed at the service provider as fast as possible. Providing the service locally at the service consumer (if possible) will result in high responsiveness in most cases. This is because the network communication is not required in such cases, hence, the performance only depends on the computing resources of the service provider. If multiple service consumers are consuming the same service, the service overlay must select a service provider such that the service is highly responsive to all connected service consumers. For instance, voice transmission using a voice chat service should have mouth-to-ear delays of at most 400 ms from the speaking service consumer to all listeners [93].

Minimum costs. Costs are specified as the resource consumption for maintaining the service overlay functionality and for providing a service. In terms of access methods of the service overlay, costs occur during the process of adding new services to the system. Furthermore, service queries can introduce high communication costs. For both methods, the costs must be minimal. In terms of service provision, costs result from the network connection between service consumer and service provider and from the utilization of computing resources at the service provider. Communication costs consider the bandwidth utilization on the network link between service consumer and service provider. Depending on the distance and intermediate network nodes between them, communication costs may vary. The utilization of computing resources is to be quantified relatively to the resources available on the service provider. Hosts with high resource capacities can provide more services or resource intensive services compared to hosts with low resource capacities. This fact must be considered when minimizing the costs for service provision. In our example of the voice chat service, minimum costs are achieved if bandwidth consumption on all involved network links is minimal.

Efficiency. The quality aspects performance and costs are tailored to achieve high responsiveness and low resource consumption, respectively. However, both aspects are important at the same time. Efficiency describes the ratio of performance and costs. High efficiency is achieved when services are provided with high performance and with low costs. Therefore, efficient services are highly responsive services consuming little bandwidth and utilizing only few computing resources on the service provider. The same applies to the access methods of the service overlay.

Robustness. Robustness of a service overlay is specified as the persistence of its functionality in case of drastic changes in the system. Especially the mobility of participants results in a high probability of connection loss. The service overlay must be able to handle such events to avoid low performance or even unavailability of services. Predicting such events and initiating countermeasures upfront can be effective and will most probably result in a more robust system providing high quality services.

Scalability. Scalability of a service overlay is defined as the ability to support large numbers of network nodes (horizontal scaling) and services (vertical scaling). This support must consider the quality aspect efficiency. Constant efficiency throughout both scaling dimension can be an indicator for a scalable system. In harsh environments, the number of participating nodes can increase dramatically. For example, in large scale incidents, such as the 9/11 attacks, 10,000 and more rescue workers can be involved in the rescue mission. They will most probably add many different services to the service overlay, they search frequently for services, and they invoke multiple service instances. The service overlay must be able to handle this high demand and still maintain a high efficiency.

2.3 Coarse System Architecture

We defined functional requirements for service provision in general and deduced non-functional requirements specific to harsh environments. This enables us to design a coarse system architecture of our proposed service overlay. Our system has a modular design. The individual system components provide the functionality identified in the functional and non-functional requirements described above. Figure 4 depicts our proposed coarse system architecture including all system components.

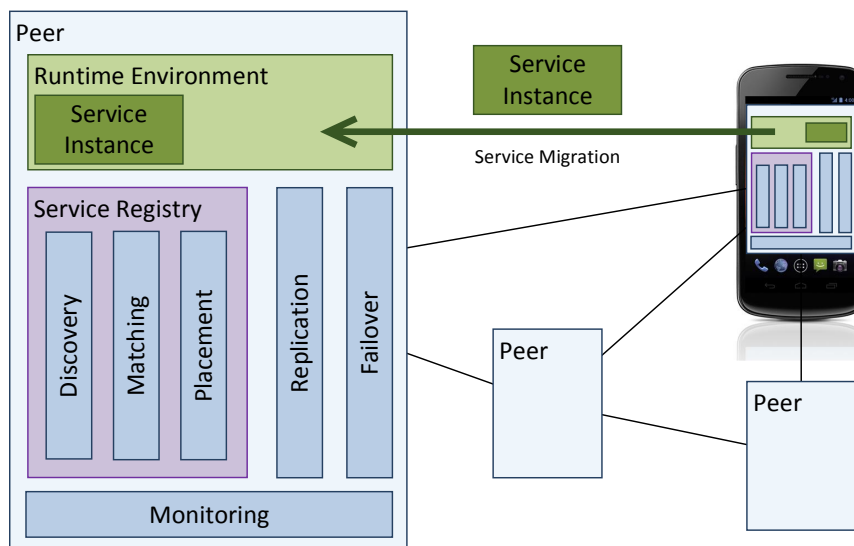


Figure 4: Coarse system architecture of a service overlay

The fulfillment of Requirement 2.1 and Requirement 2.2 requires a network of devices. These devices all can act as service provider and service consumer at the same time. Such a behavior is also known from peer-to-peer systems where every peer can act as client and server simultaneously. Therefore, we base our system architecture on the peer-to-peer paradigm and use the same terminology. Interconnected devices are called *peers* and the entire network (the basis for service provision) is called a *peer-to-peer network*.

In general, peer-to-peer systems can be classified into *unstructured* and *structured* peer-to-peer systems. In unstructured systems, objects (e.g., files, services, etc.) are accessed by flooding the network with search queries. The object owner or any other peer that knows about the location of the requested object responds to the query. In structured systems, objects are addressed by their identifier (i.e., ID, hash key, etc.). Using a routing algorithm, requests are routed to a peer associated with the object identifier. This peer then responds to the request. Finally, *hybrid* peer-to-peer systems exist that combine both concepts.

Our system design of the peer-to-peer service overlay is not tailored to a specific class of peer-to-peer system. The final version depends on the various system components and their implementation. In this thesis, we focus on a hybrid peer-to-peer system. Addressing services and peers by their identifier is a useful feature. However, searching for services by flooding search queries through the network might be useful, as well. A strong orientation towards either structured or unstructured systems is, therefore, inappropriate at this point.

Runtime Environment: Providing services means executing software applications that in return provide their functionality to service consumers. To execute software services, we propose that peers have a runtime environment. This runtime environment can be understood as a middleware for service provision. Although we do not formulate security as a requirement of our proposed service overlay per se, the runtime environment can cope with some security-related issues. It encapsulates the services during their execution and it isolates them from the peer's hardware and software as well as from other services on the same peer. If done properly, this can protect the peer and other services from malicious code. In principle, the runtime environment can be achieved by the concept of virtual machines.

Service Registry: To fulfill Requirement 2.3, the peer-to-peer system must be capable of adding new services and of searching and accessing existing services. To meet this requirement, we propose a service registry component that implements a consistent interface to access these methods. To fulfill the identified requirements for harsh environments, the service registry is required to be a more complex component. It consists of further sub-components (cf., Figure 4): the service discovery, the service placement, and the matching components. These sub-components then implement parts of the functionality described in the requirements 2.3, 2.4, and 2.5.

Service Discovery: To add and search for services in the peer-to-peer service overlay, the service discovery component is necessary. Participants can register their (new) services and, thus, make them available to other participants in the network. To search for a specific service, the service discovery component accepts search queries from participants and responds with an appropriate software service that provides the requested functionality. Ideally, this response contains the network address of the peer executing the software service rather than the service coder or the executable binary file. If the service is not executed anywhere in the service overlay, the discovery can trigger the instantiation process. Finally, the chosen service discovery approach for this component in the final system implementation has the greatest impact on the class of peer-to-peer systems the service overlay falls into.

Matching: Earlier, we defined the software and hardware properties of devices carried by the participants (cf., Definition 2.2). Peers directly map to these devices, hence, they inherit the software and hardware resources including, for example, operating system, CPU, memory, etc. In harsh environments, the devices and their resources are highly heterogeneous. Hence, the software and hardware requirements of the service must be considered when instantiating a software service (cf., Requirement 2.4). Only peers with matching software and hardware resources can execute that particular service. Therefore, a matching component is required handling this process of the initial service placement.

Service Placement: The third component of the service registry is the service placement component. It is responsible to select peers for service provision with highest service quality possible. This is not to be confused with the matching component which identifies peers in general capable of executing a given software service. From the set of matching peers (identified by the matching component), service placement selects a peer that provides a given service with highest efficiency. Therefore, this component also considers the quality aspects performance and costs. In a large and dynamic network, the locations of service consumer and service provider have an impact on the utilization of network resources and the quality of provided services. If only one service consumer is interested in one particular service, the most efficient location for service provision is his own local device (provided his device is capable of executing the software service in the first place). If more service consumers are interested in that service, the most efficient location for service provision will most probably differ. Especially in harsh environments where the network is exposed to high dynamics, the placement component must continuously relocate the software service to provide high service quality.

Service Migration: To achieve the relocation of services, it is of great importance that software services can be migrated between peers during their execution. A running service has an execution state which together with the service code or binary file also has to be transferred to the new service provider. However, frequent service migrations might directly influence the accuracy of the service discovery as well as the performance of the service. Especially if the executed software service is suspended during the migration process, it cannot provide its functionality to service consumers. Therefore, these interruptions must be eliminated in a real world implementation of a service overlay.

Replication and Failover: To target the quality aspect robustness, we need mechanisms that handle failures of service providers and services to minimize the number and the durations of interruptions during service consumption. To achieve this, we propose the integration of the replication and failover components. While the replication component is responsible for replicating services as a backup, the failover component handles failures of service providers and services. Together, both components ensure that service consumers can seamlessly consume their services without noticing failures in the system.

Monitoring: Finally, all these components need different types of information to provide their functionality. This information includes number and status of active peers, available hardware and software resources on the peers, location and status of software services. This information is provided by the monitoring component. It gathers necessary information and provides it to the other components.

We call a system that consists of these components to provide versatile software services in a peer-to-peer manner a *peer-to-peer service overlay*. In the next section, we formalize the

functionality and properties of our proposed service overlay. In Chapter 5, we then provide more details on the system architecture and on selected components of the service overlay.

2.4 Formal System Model

We described harsh environments and the underlay network, deduced requirements for service provision in such environments, and presented a coarse system architecture for a peer-to-peer service overlay. In the present section, we further deepen the understanding of the peer-to-peer service overlay. We present formal definitions of the peer-to-peer service overlay. Focusing on disaster relief, we also describe special properties of individual components, if applicable. The present section is based on our chapter “A Formal Model for Peer-to-Peer Systems” [154] in “Benchmarking Peer-to-Peer Systems” [60].

By logically interconnecting underlay hosts with each other, the overlay network is formed. Participating hosts are called peers and the network is called a peer-to-peer overlay network. Although peers are (logically) connected with each other in the peer-to-peer overlay network, a direct physical connection between them in the underlay network is not required. The communication path in the underlay network between both peers might consist of several hosts (routers) and the connections between them. Messages exchanged between both peers are then forwarded by these underlay hosts or routers. Attributes of peers and overlay connections depend on attributes of the corresponding hosts and of the physical underlay connections between those hosts, including all routers in the communication path.

Definition 2.8. A **peer-to-peer overlay** is a directed, loop-free Graph $\mathcal{O} = (P, L)$ on top of an underlay network $\mathcal{G} = (V, E)$ with $P \subseteq V$ the set of participating peers $P = \{p_1, \dots, p_n\}$ and the overlay connections $L \subseteq \{(p, q) | p, q \in P, p \neq q\}$ between peers. The total number of peers in the overlay network is given as $n = |P|$. A peer $p \in P$ is bound to a host $v \in V$ which can be simplified as $p = v$. An overlay connection $(p, q) \in L$ with $p, q \in P, p = v_0, q = v_l$ and $v_0, v_l \in V$ is characterized by a path v_0, \dots, v_l in the underlay of length l with $(v_0, v_1), \dots, (v_{l-1}, v_l) \in E$. Depending on the considered peer-to-peer system, $\forall (p, q) \in L : (q, p) \in L$ holds.

For many metrics, functions, and algorithms used in peer-to-peer systems, the neighbors of a peer are of interest. Therefore, a definition of a peer’s neighborhood is needed:

Definition 2.9. A **Neighborhood** $N(p) \subseteq P$ of a peer p is defined as the set of peers p is directly connected to in the overlay: $N(p) = \{q | (p, q) \in L\}$. A peer $q \in N(p)$ is called a neighbor of peer p .

Hosts and underlay connections have attributes modeling their resource capacities and connection parameters, respectively. Since a peer is bound to a specific host, the peer directly inherits the resource capacities of that host. On the other hand, overlay connections are communication paths (indirect connections) in the underlay network. Therefore, the connection parameters of an overlay connection between two peers depend on the parameters of single links of the underlay communication path. Hence, a non-negative cost function is needed, defining the costs of an overlay connection with respect to the connection parameters of the corresponding path in the underlay:

Definition 2.10. Let \oplus be an operation $\oplus : \mathbb{R}_+^{|W|} \times \mathbb{R}_+^{|W|} \rightarrow \mathbb{R}_+^{|W|}$, i.e., component-wise addition. The **cost function** $c : L \rightarrow \mathbb{R}_+^{|W|}$ assigns each overlay link $(p, q) \in L$ the non-negative costs $c((p, q)) = \bigoplus_{j=1}^l w((v_{j-1}, v_j))$ with respect to the path in the underlay network $(v_0, v_1), \dots, (v_{l-1}, v_l) \in E$.

As discussed, we assume a peer-to-peer overlay network for harsh environments to consist of static hosts in the infrastructure and of mobile hosts, such as smart phones or laptops. Mobile hosts are exposed to high dynamics due to their movement. We model this case as a fully meshed graph with dynamic link attributes. Dynamic attributes model time-dependent link properties. The same modeling approach is used to model the arrival and departure of peers to the system, called *churn*. We model the participation of peers and the availability of links as follows:

Definition 2.11. For any point in time $t \in T$, $P(t)$ is defined as the subset of all **participating peers** $P(t) \subseteq P$ at time t . Also, $L(t)$ is defined as the subset of all overlay links $L(t) \subseteq L$ with $L(t) \subseteq \{(p, q) | (p, q) \in L, p, q \in P(t), p \neq q\}$ at time t . Additionally all overlay link costs $c((p, q), t)$ are dependent on time t .

Peers in the overlay network must be identifiable and addressable to exchange messages with other peers. Also, in many cases addresses are needed to assign certain responsibilities to individual peers.

Definition 2.12. To address individual peers in an overlay network the peers are assigned identifiers from an identifier space: $id_p : P \rightarrow \mathbb{N}_0$. Also, a **distance function** for two neighboring peers p and q has to be defined as a function $d : P \times P \rightarrow \mathbb{R}_+$.

This helps to define the routing in peer-to-peer overlays. Routing is a function that can be described as the process of (dynamically) finding an (efficient) path from peer p to peer q . A prominent example for routing is the greedy routing algorithm for structured peer-to-peer systems. Greedy routing in the overlay from a peer p to another peer q is defined as the step-wise forwarding of a message at peer u with the following rule: Using the distance function $d(x, q) = |id_p(q) - id_p(x)|$ forward the message to a neighbor $v \in N(u)$ with $\forall y \in N(u) : d(v, q) \leq d(y, q)$.

Peer-to-peer systems usually serve applications that implement storage, access, and exchange of data objects. In this thesis, we focus on service objects and service instances being accessed by peers to harness versatile services. We define the notions of service objects, service instances, and associated functionality as follows:

Definition 2.13. Service objects $O = \{o_1, \dots, o_k\}$ are special types of stored (data) objects. They are represented by executable data items and, moreover, they also can be invoked.

Definition 2.14. Invoked and, thus, currently running service objects are called **service instances** $I = \{i(o_1)_1, \dots, i(o_k)_m\}$. During execution, service instances provide the designated functionality to accessing peers.

To handle service objects, their storage, and their invocation inside the peer-to-peer overlay, identifiers from an identifier space are assigned to service objects. This is related to the assignment of identifiers to peers.

Definition 2.15. In a peer-to-peer overlay, the service objects are assigned identifiers from an identifier space: $id_O : O \rightarrow \mathbb{N}_0$. Service objects are assigned to peers according to the service objects' identifiers using an **assignment function** $\alpha : \mathbb{N}_0 \rightarrow P$.

Every service object can be invoked multiple times, thus, creating multiple service instances. Each running service instance must be (uniquely) identifiable and addressable. Therefore, we introduce an identifier assignment function for service instances, too:

Definition 2.16. Service instances are assigned identifiers from an identifier space: $id_I : I \rightarrow \mathbb{N}_0$. Service instances are placed on peers according to a **placement function**: $\rho : \mathbb{N}_0 \rightarrow P$.

Often, it is beneficial to use the same identifier space for service objects, service instances, and peers. Routing a request to a service object o [or to a service instance $i(o)$] in a service overlay requires the definition of $q = \alpha(id_O(o))$ [or $q = \rho(id_I(i(o)))$] with the assignment function α [or placement function ρ] and executing the routing rule defined by the respective peer-to-peer system.

With the above definitions of service objects and service instances we now can define the notion of peer-to-peer service overlays:

Definition 2.17. Specific peer-to-peer systems that provide functionality to invoke service objects and to access service instances are called **service overlays**.

Handling service objects in a peer-to-peer service overlay requires two basic methods in the service registry: *put* and *invoke*. Both methods take a service object (reference) as input parameter. In the following, the peer $q = \alpha(id_O(o))$ is the peer assigned to the service object o (the target peer), and peer p is the peer applying the respective method with service object o :

- The method *put* is defined as registering a service object o at peer q . Peer q then knows about the existence of service object o at peer p and also stores all other locations of service object o .
- The method *invoke* is defined as contacting peer q and requesting access to a service instance $i(o)$ of the service object o . If no service instance exists, peer q initializes the service instance $i(o)$ of service object o and registers it. Finally, peer q transmits the set of locations for all service instances $i(o)$ back to peer p . Peer p can then access the service instance $i(o)$.

Handling service instances in a peer-to-peer service overlay requires two additional methods in the service registry: *register* and *access*. Both methods take a service instance (reference) as input parameter. The peer $r = \rho(id_I(i(o)))$ is the peer the service instance $i(o)$ is placed on for execution. Peer r is also the peer applying the *register* method for service instance $i(o)$. Peer p is the peer applying the *access* method for service instance $i(o)$. Peer $q = \alpha(id_O(o))$ is again the peer assigned to the service object o of consideration.

- The method *register* is defined as registering a service instance $i(o)$ by informing peer q about the existence of service instance $i(o)$ at peer r . Peer q then stores the locations of service instances $i(o)$ of the service object o .

- The method *access* is defined as connecting to the service instance $i(o)$ at peer r to consume the provided service.

Figure 5 depicts a more detailed system design of our proposed service overlay. The figure shows the important methods of the individual components and how the components interact with each other. Every peer in the service overlay implements this system design.

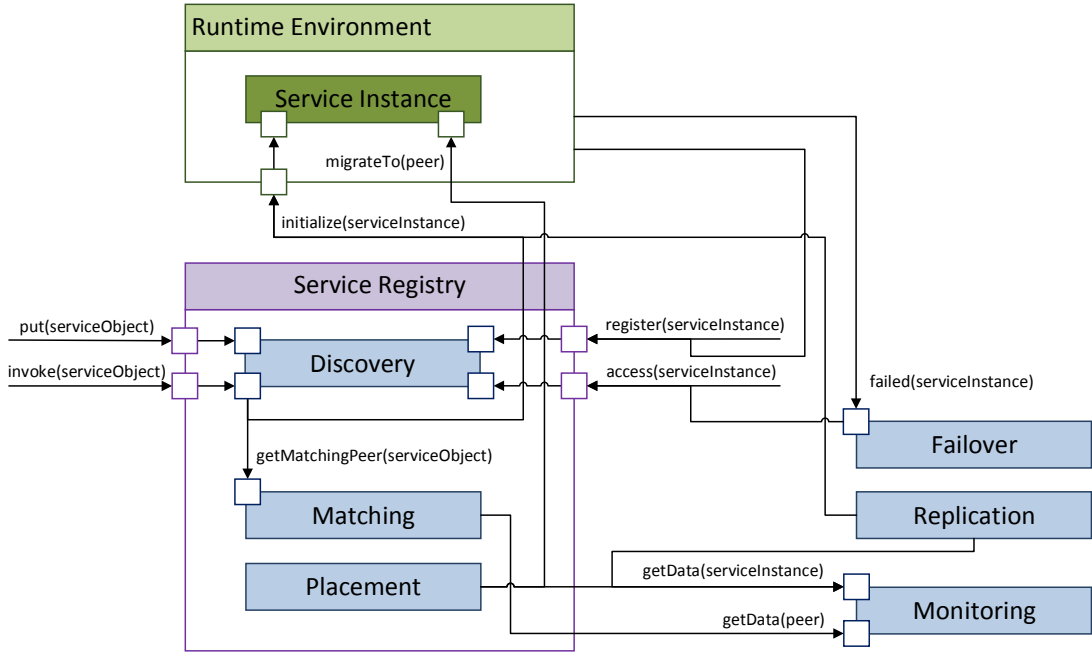


Figure 5: System design of a service overlay on one peer

2.5 Summary

In this chapter, we gave an overview on the entire system consisting of the environment, the service overlay, and simulation models. We defined properties of the three parts that make for a harsh environment: the fluctuating density and movement of participants, the overloaded or destroyed communication infrastructure, and the obstructed or destroyed geographical region. Furthermore, we defined the underlying physical network that is formed by the participants and the infrastructure: the underlay network. As an example for application scenarios in harsh environments, we use disaster recovery, since environmental constraints and requirements are tighter in such scenarios than in mobile-assisted public gatherings, for example.

Furthermore, we deduced functional and non-functional requirements resulting from the definition of harsh environments and the task of service provision in such environments. From these requirements, we identified necessary system components for service provision and presented a coarse system architecture. We base our system on the peer-to-peer concept extending it with components for service provision, consumption, discovery, placement,

replication, and system monitoring. Finally, we presented a formal system model of the service overlay on top of the underlay network. We further defined functions and methods a service overlay must implement to provide services in general and especially in harsh environments.

3 Related Work

We introduced the topic of this thesis in Chapter 1. We formulated the problem of service provision in harsh environments and outlined the contributions of this thesis. In Chapter 2, we presented an overview of the overall system: the harsh environment, the service overlay, and the simulation models. We defined the properties of harsh environments: obstructed geographical region, overloaded or destroyed communication infrastructure, and fluctuating density and movement of participants who, at the same time, interact with the service overlay. We deduced requirements for service provision in general and for harsh environments in particular. Finally, we presented a coarse system architecture and defined important terminology as well as system properties of our proposed service overlay.

In the present chapter, we investigate related work for the three major research fields of this thesis: the communication infrastructure in harsh environments, peer-to-peer-based service provision, and simulation models. First, we give an overview on communication infrastructures in Section 3.1. This also includes basic technology and approaches in the underlay network proposed for disaster recovery missions. Second, we present relevant approaches for service provision in general and in disaster recovery in particular (Section 3.2). We also focus on approaches to individual system components as described in Section 2.3: service matching, service placement, service discovery, replication and failover, and monitoring components. Thereby, we put our emphasis on the two components that are most important for the functioning of the service overlay in harsh environments: service placement (Section 3.2.3) and service discovery (Section 3.2.5). Third, we give an overview of relevant simulation models in Section 3.3. This includes the mobility of participants and their communication behavior but also the chosen simulation framework for our investigations.

3.1 Communication Infrastructure

In the present section, we give an overview on communication infrastructures for harsh environments. We mainly focus on communication infrastructures and communication systems for emergency response to target the example application scenario introduced in Section 2.1. However, most of the concepts presented below are also generally applicable to harsh environments. We classify infrastructure concepts into backbone-based and decentralized concepts. We further classify decentralized infrastructure concepts into roll-out and pre-deployed concepts. An overview of our classification of communication infrastructure concepts for harsh environments is depicted in Figure 6.

We discuss backbone-based infrastructure concepts in Section 3.1.1 and decentralized roll-out infrastructure concepts in Section 3.1.2. We present pre-deployed decentralized infrastructure concepts in Section 3.1.3. Our proposed concept to support the communication infrastructure in harsh environments falls into the last class of concepts. Hence, we put our emphasis on this class during our evaluation. Furthermore, decentralized concepts usually require a routing protocol to transmit data packets between participants. We investigate

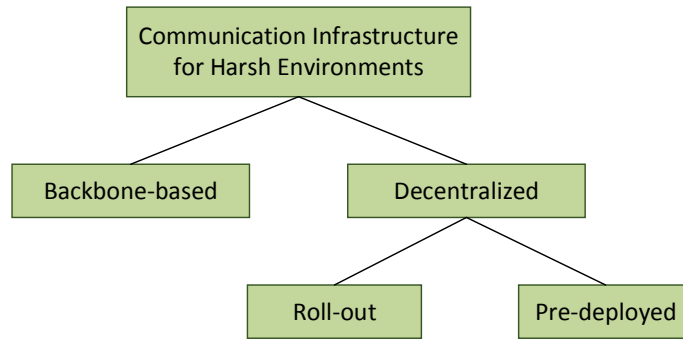


Figure 6: Overview of communication infrastructure concepts for harsh environments

suitable routing protocols in Section 3.1.5. To evaluate the infrastructure concepts in the three classes introduced above, we specify the following five requirements.

- R.I1: No/short deployment time:** The majority of survivors are found within the first hours after a disaster has struck. Hence, the communication infrastructure must be readily available after a disaster so that rescue workers can start with their rescue mission immediately. Setting up the communication infrastructure can consume valuable time.
- R.I2: Resilient to failures:** Especially in disaster situations, the communication infrastructure can be subject to severe failures. These failures might result from disaster aftermaths, but in more general scenarios also from communication overload or even cyber-attacks. The communication infrastructure must still provide communication service even if parts of the infrastructure fail.
- R.I3: High throughput:** To provide versatile software services, high network throughput is required. Especially media services, such as voice communication, picture sharing, or video streaming, require high bandwidth from the underlay network.
- R.I4: Widely available:** Primarily, the communication infrastructure must be available in locations where many people live, i.e., cities. However, the communication infrastructure must also be available in rural areas. In case of a disaster, rescue workers must be able to use the infrastructure where ever they have to work.
- R.I5: Use existing standards:** The communication infrastructure should be based on existing standards and, if possible, also on open standards. If the infrastructure is based on widely used communication technology, more participants are able to harness the infrastructure without the need for special hardware. Also, open standards can help assisting the infrastructure with additional devices in case of failures more easily, because such devices are usually cheaper and easier to acquire.

3.1.1 Backbone-based Infrastructure

The typical method to deploy a communication infrastructure is to install fixed backbone nodes in the region to support the communication. These nodes can be cell towers as

known from the GSM and UMTS standards used for mobile phone communication. Provided such a pre-installed backbone-based infrastructure was not damaged by the disaster, this infrastructure has no deployment time after a disaster has struck. Hence, it is readily available and rescue workers can start directly with their rescue mission. However, it takes long planning and installation phases upfront to deploy such a backbone-based infrastructure. Furthermore, deployment and operation of large cell towers are connected with high costs. The Terrestrial Trunked Radio (TETRA) standard implements the backbone-based concept and is used as the standard communication technology for disaster relief in Europe.

Terrestrial Trunked Radio (TETRA)

Today, professional digital voice and data communication is based on the TETRA standard in most European countries [180]. TETRA is based on cell type system deployments and was specifically designed to be used by professional users, such as government agencies and emergency departments. The TETRA standard defines a digital mobile radio system in the ultra-high frequency (UHF) band below 1 GHz and was released by the European Telecommunication Standards Institute (ETSI) in 1995. In Germany, the frequencies 380–385 MHz (uplink) and 390–395 MHz (downlink) are reserved for emergency services using TETRA. It is a highly reliable communication system that provides instant transmission, high availability, and security. Data transfer is efficient and long range (several km), however, slow compared to the WiFi standard, for instance. With the latest version of the TETRA standard, bandwidths of up to 691.2 Kbit/s can be achieved.

Although the TETRA standard provides many advantages for the use in disaster situations, it also has several drawbacks regarding the requirements specified above. First, the relatively low bandwidth supported by the TETRA standard can degrade the quality of services with high bandwidth demands (e.g., video streaming). Second, the cell-based concept of such an infrastructure is vulnerable to failures. If a TETRA cell tower is destroyed by a disaster, communication services might be degraded or even fail completely. However, there is still the possibility to deploy small temporal cell towers in regions where no TETRA cell tower is available. Finally, concerns have been raised about health issues connected with TETRA cell towers [145].

3.1.2 Decentralized Roll-out Infrastructure

During the last decade, a plethora of decentralized communication systems for disaster relief has been proposed [21, 35, 42, 50, 123]. Most of these systems are infrastructure-less and incorporate the peer-to-peer paradigm with wireless ad-hoc connections based on the IEEE 802.11 standard [90]. The IEEE 802.11 standard uses 2.4 GHz and 5 GHz bands and provides two modes of operation: infrastructure and ad-hoc mode. In infrastructure mode, devices connect to a central router to access the Intranet and the Internet. In ad-hoc mode, WLANs are created by interconnecting wireless devices directly without intermediate routers. In mobile ad-hoc networks the wireless devices have the ability to move, thus, changing the network topology. Topology changes happen because wireless devices have a limited transmission range (up to 300 meters under good conditions). By moving, devices leave the transmission range of other devices and connect to devices that come into transmission range. In general, mobile wireless ad-hoc networks configure and organize themselves

autonomously. Depending on the used WiFi standard, such networks can achieve bandwidths of up to 300 Mbit/s with today's most commonly used hardware. Extensions to the WiFi standard are in progress to further increase the bandwidth.

To assist such a decentralized mobile communication system, additional infrastructure can be rolled out on demand. For this purpose, usually wireless mesh routers are used. Rescue workers deploy these routers on strategic locations in the area. The routers then form a wireless mesh network that can be used for communication over long distances. Wireless mesh networks are special ad-hoc networks where wireless devices are typically stationary and distributed over a particular area [9]. The visual representation of the resulting network topology has strong similarity to a mesh, hence, the name mesh network. Wireless mesh networking is already part of many Linux distributions in the form of the IEEE 802.11s amendment [86]. In typical mesh networks, there are two types of devices: the mesh clients and the mesh routers. The mesh routers form the wireless mesh network by connecting to each other via ad-hoc wireless connections. Certain mesh routers can also serve as Internet gateways. Mesh clients, on the other hand, connect to mesh routers and use the created mesh infrastructure to send and receive messages. Although mesh routers are responsible for routing messages through the network, mesh clients may also participate in the routing process [10] (cf., Section 3.1.5 for details on routing protocols).

Compared to the backbone-based infrastructure concept, a decentralized concept has no single point of failure. If individual mesh routers fail, the overall network can still operate in most cases. Furthermore, roll-out infrastructures can be used anywhere they are needed. They are very flexible and can be deployed and removed quickly (as compared to the process of installing backbone nodes). However, if such an infrastructure is used in case of a disaster, the deployment time in the first hours after the disaster can have a negative effect on the recovery mission [25, 179]. Thus, mesh routers need to be small and light weight. Furthermore, the routers must configure themselves automatically to communicate with the other mesh routers and with devices of first responders, as well [179]. Finally, the locations where to place these routers have to be chosen such that the resulting network provides reliable communication service [59, 161]. Reina et al. proposed an evolutionary algorithm to reposition the mesh routers to increase the network connectivity [161].

DUMBONET

Kanchanasut et al. proposed DUMBONET, a combination of mobile ad-hoc networks and satellite IP networks [102]. It is designed for collaborative emergency response operations in multiple disaster-affected areas. In their evaluation, they used elephants equipped with wireless mesh routers to support first response communication. Although this idea seems funny at first, equipping vehicles and working animals with networking hardware can help in disaster situations. Especially, if they are used to move between multiple disaster areas.

Rapid deployment of wireless mesh networks for disaster relief is important for the success of rescue missions [140]. However, deploying this infrastructure consumes time and personnel both of which are restricted resources in disaster situations. Also, this requires special hardware introducing further costs. We believe such an infrastructure should be readily available directly after the disaster to further reduce reaction time.

3.1.3 Pre-deployed Decentralized Infrastructure

Besides rolling out additional infrastructure nodes on demand to create a wireless mesh network, the communication infrastructure can also be pre-deployed. Similar to the backbone-based concept, the wireless mesh routers are then installed on strategic locations as a precaution before a disaster strikes. In case of a disaster in that particular area, the infrastructure would be readily available. Disaster-affected areas that are not covered by such a network would be connected through additional roll-out infrastructure.

A pre-deployed decentralized infrastructure would be beneficial in regions where many people live, i.e., cities. In such regions, the infrastructure would be used for more than just disaster relief communication. Inhabitants could use the infrastructure for Internet access, for instance. Below, we present three prominent city-wide wireless mesh and community networking projects to be evaluated as supportive infrastructure in harsh environments. In general, such city-wide WiFi networks are also called municipal wireless networks and are primarily used to provide city-wide wireless Internet access.

MIT Roofnet project

One of the first experimental IEEE 801.11 mesh networks was the Grid Roofnet project on the campus of the Massachusetts Institute of Technology in Cambridge, MA [47]. The project started as a master's thesis and developed into a production quality wireless ad-hoc network [6]. The resulting network provided broadband Internet access to users in Cambridge. In 2003, it consisted of over 35 nodes and covered an area of about two square kilometers. Each node of the Grid Roofnet project was a PC equipped with the Linux operating system and two network interfaces: a wired network interface and a wireless network interface based on the IEEE 802.11b standard. The wireless network interface was connected to an omnidirectional antenna mounted on the roof of the building. These nodes were installed in several buildings of the MIT campus in Cambridge. They were interconnected, thus, creating the wireless mesh network from roof-to-roof. Several gateway nodes provided Internet access for the entire network.

The project used a proprietary routing protocol called SrcRR, based on the Dynamic Source Routing protocol (DSR) [99]. As opposed to the DSR protocol, SrcRR uses the Expected Transmission Count (ETX) metric to determine a route between two nodes. Access and transport of media is handled by the Extremely Opportunistic Routing (ExOR) algorithm [31]. Users were able to connect to each node and obtained an IP address to access the Internet. IP addresses were not routable, thus, network address translation (NAT) was used. A NAT allows multiple clients to access the Internet through a gateway node using only one routable IP address. This also means that users connected to different Grid Roofnet nodes were not able to connect and communicate with to each other through the Grid Roofnet network.

The main goal of the Grid Roofnet network was to create a self-organized network for Internet access. The network was rather small and the project used specialized hardware to reach the goals. However, the project showed that mesh networks can cover a large geographical area with only few nodes if they are mounted on roofs.

Google WiFi

In Mountain View, CA, Google Inc. deployed a city-wide WiFi network for Internet access¹. The network consists of over 500 Tropos MetroMesh wireless routers. The routers are mounted primarily on city owned utility poles (i.e., light poles). Figure 7 shows the distribution of the Google WiFi routers in the city of Mountain View, CA². Google employees, Citizens, and visitors can access the Internet via this mesh network. The service started in August 2006 and requires users to have a Google account. Connections can be established to the service set identifiers (SSID) “GoogleWiFi” (which has no encryption) or “GoogleWiFiSecure” (providing WPA encryption). Google limits the data transfer rates to 1 Mbit/s per client.

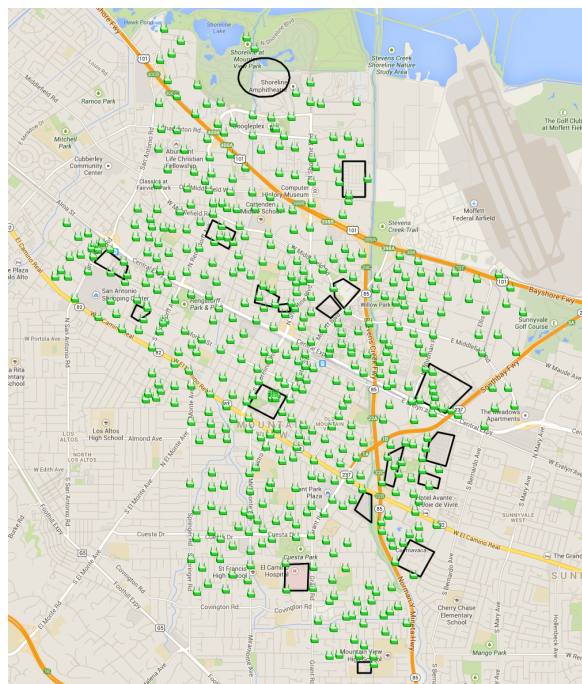


Figure 7: Coverage Map of Google WiFi in Mountain View, CA (Screenshot)

In contrast to the Grid Roofnet network in Cambridge, NAT is not used in the Google WiFi network. Hence, every client receives a routable IP address from the DHCP Server. The IP address has a DHCP lease of one hour. The network uses a proprietary Tropos routing algorithm. There are three distinct wired gateways to access the Internet. All Internet traffic is routed to one of these three gateway nodes.

Afanasyev et al. measured the Google WiFi network in spring 2008 [4, 5]. Their findings about the mesh connectivity revealed that only 5% of the routers had a unique neighbor. The median router had at least four neighboring routers and 10% had more than eight neighbors.

¹ Google WiFi website available online at <http://wifi.google.com/>. The “Google WiFi Terms of Service” are available at <http://wifi.google.com/terms.html> (visited on April 3, 2014)

² Screenshot taken from the “Google WiFi Mountain View Coverage Map” website: <http://wifi.google.com/city/mv/apmap.html> (visited on April 3, 2014)

As of spring 2008, the network had between 1000 and 2500 simultaneously active users during the course of a single day. They produced roughly 500 Gbytes of data traffic. Further details of the network measurements can be obtained from the works of Afanasyev et al. [4, 5]. In 2007, Arjona and Takala from Nokia Networks investigated the Google WiFi network on its VoIP capabilities [13]. Their findings indicate the network to provide good link quality only in few spots, mainly in the downtown area. They state that the current router density is not sufficient for proper support of voice services.

As of the time writing this thesis, Google announced on their Google WiFi website (on April 3, 2014) to take down the WiFi network in Mountain View. They stated that the network would operate for 60 days (roughly until early May). They are planning to install a new outdoor WiFi network in downtown Mountain View, along Castro Street.

Freifunk

“Freifunk” networks are community driven networks provided and maintained by owners of wireless routers in cities and villages in Germany³. The Freifunk community is part of a global movement for free communication infrastructure and open frequencies. Their main goal is to provide a city-wide wireless mesh network to share files and services among citizens. Furthermore, Internet access can also be obtained through the Freifunk network provided community members share their Internet connection with the Freifunk community. The most prominent and largest Freifunk networks in Germany are established in Berlin, Hamburg, Lübeck, and Leipzig, among others.

To join the Freifunk network, a user has to download a special router firmware called OpenWrt⁴. This firmware is based on the Linux operating system and is developed and provided by the Freifunk community for free. Besides normal WiFi routers, community members also install outdoor routers on their houses to increase the link quality and network range. In November 2006, measurements of the Freifunk network in Berlin revealed the network to have 316 nodes on average, varying between 199 and 416 [125, 126]. The number of links varied between 291 and 951 (633 on average). For routing, the network used OLSR [55] for a long time. Today, the Freifunk community claims that the B.A.T.M.A.N. routing protocol [137] is increasingly used in their communities and achieves higher performance than OLSR.

As opposed to the Google and Cambridge WiFi networks, the Freifunk network is based on voluntary work and resource sharing. Community members all over Germany prove that there is an interest in a free and city-wide wireless community network. Sharing files, services, and Internet connectivity with others to increase the own network range in return is one of the main reasons to join such networks. However, the router firmware is not available for every router on the market. Also, installing the firmware requires deep technological skills in many cases. This might be the main reason for the rather small number of participants in a city as big as Berlin.

³ Freifunk community website (visited on April 4, 2014): <http://www.freifunk.net/>

⁴ OpenWrt - Wireless Freedom website (visited on April 4, 2014): <http://www.openwrt.org/>

3.1.4 Results

We summarize our findings above in Table 1. All concepts are based on existing standards. The backbone-based concept relies on the TETRA standard and decentralized concepts use the IEEE 802.11 WiFi standard. The backbone-based communication infrastructure has no deployment time (but long planning and installation phases before the disaster), is usable over a large area, and can be further supported with temporal roll-out infrastructure. However, the TETRA standard is relatively old and does not provide high throughput for data transmission. Furthermore, the cell-based TETRA system is prone to failures since communication partly relies on individual cell towers.

In contrast, decentralized infrastructure concepts are more resilient to failures. If individual nodes fail, only small parts of the area are affected due to the limited range of the WiFi mesh routers. Depending on the chosen WiFi standard, high throughput can be achieved. Rolling out such a decentralized infrastructure always comes with the drawback of the deployment time. Pre-deploying the infrastructure, on the other hand, limits the network coverage to the area where the nodes were installed previously. Hence, a combination of pre-deployed and decentralized roll-out infrastructure is the best fit for disaster relief.

	No/short deployment time	Resilient to failures	High throughput	Widely available	Use existing standards
Requirements	R.I1	R.I2	R.I3	R.I4	R.I5
Backbone-based					
TETRA [180]	✓	—	—	✓	✓
Decentralized roll-out					
DUMBONET [102]	—	(✓)	✓	✓	✓
Pre-deployed decentralized					
MIT Roofnet [47]	✓	(✓)	(✓)	—	✓
Google WiFi	✓	(✓)	✓	—	✓
Freifunk	✓	(✓)	✓	(✓)	✓
Our proposed approach					
CityMesh	(✓)	✓	✓	(✓)	✓

Table 1: Comparison of presented infrastructure support approaches for disaster recovery

In Chapter 4, we propose to use existing public and private wireless routers in urban areas to support emergency communication with a city-wide mesh network: the CityMesh. Our approach would eliminate deployment time for inhabited areas, i.e., cities. The network is decentralized, provides high throughput and covers large areas in cities with high density. The high density makes it further resilient to individual node failures. However, this approach requires regulation by law and additional precautions, such as firmware updates and hardware upgrades for existing routers.

In the next section, we investigate underlay routing protocols for harsh environments.

3.1.5 Routing

Routing is the process of (dynamically) finding an (efficient) path from a sender node to a source node. Wireless ad-hoc networks require special routing protocols due to the high dynamic introduced by the mobility of network nodes. Today, a plethora of routing protocols for wireless ad-hoc networks exists. In general, these protocols can be classified into *proactive*, *reactive (on demand)*, and *hybrid* routing protocols. For more details on routing protocols for various application scenarios the reader is referred to several surveys on ad-hoc routing protocols [2, 3, 8, 101].

To evaluate existing routing protocols, we first specify two general requirements for routing protocols (R.R1 and R.R2). Recalling the properties of harsh environments from Section 2.1, we deduce further requirements for routing protocols specific to harsh environments (R.R3, R.R4, and R.R5):

- R.R1: Low overhead traffic:** Any data traffic imposed on the network reduces the overall available bandwidth. This also includes overhead data traffic generated by the routing protocol. Hence, the overhead data traffic of the routing protocol should be small in general.
- R.R2: Responsiveness:** Waiting for a route to be discovered when sending data introduces a large delay on pending packets. Therefore, the routing protocol should provide requested routes quickly.
- R.R3: Small routing information:** The underlay network in harsh environments is formed by smart devices carried by rescue workers. These devices have restricted (hardware and software) resources, hence, stored routing information must be small.
- R.R4: Support large networks:** In harsh environments, many mobile participants and static infrastructure devices create the underlay network and are continuously exchanging information. The routing protocol must be able to handle a large number of connected nodes and provide accurate routes quickly upon request.
- R.R5: Support high dynamics:** The underlay network in harsh environments is exposed to high dynamics. Participants move, thus, continuously change the network topology. The routing protocol must be able to handle these dynamics and still discover accurate routes quickly upon request.

Proactive Routing Protocols

Proactive routing protocols are also called table-driven protocols. They are based on the concept of every node continuously sharing its knowledge (routing table) about the network topology with the other nodes. However, these routing protocols produce large data traffic. Furthermore, changes in the network topology need some time until nodes incorporate them into their routing tables. If a sender node tries to send a message to a certain destination, these long response times can lead to erroneous routes and the message can be lost. Proactive routing protocols are most suitable for networks with static or slowly moving nodes.

The most prominent proactive routing protocols are the Destination-Sequenced Distance Vector (DSDV) [158] routing protocol and the Optimized Link State Routing (OLSR, specified in the RFC 3626 [55]) protocol. DSDV is effective for routing in small ad-hoc networks because it frequently floods the entire network and requires every node to maintain a complete routing table. OLSR uses selected nodes, so-called multipoint relays, to forward broadcast messages. This saves data traffic compared to the flooding approach used in DSDV. The Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) [137] and the BABEL (specified in the RFC 6126 [51]) routing protocols further reduce the amount of data traffic and routing table sizes. Hence, both protocols support larger ad-hoc networks compared to DSDV and OLSR. Real-world comparisons show B.A.T.M.A.N. and BABEL to outperform OLSR [1, 133].

Reactive Routing Protocols

Reactive routing protocols calculate routes from source to destination at the time the route is requested. They are suitable for dynamic networks with faster node movement. If a sender node wants to send a message to a certain destination node with unknown route information, it starts the route discovery process. Usually, the sender node broadcasts route discovery messages to its neighbors until the route to the destination node is found. This behavior saves overall overhead traffic that would be generated continuously by proactive routing protocols. However, if many nodes try to discover routes at the same time, the network might congest due to the large amount of data traffic generated by the route discovery messages. Furthermore, the route discovery introduces a delay to the message that the sender node tries to send.

The most prominent reactive routing protocols are the Ad-hoc On-Demand Distance Vector routing (AODV, specified in RFC 3561 [157]) and the Dynamic Source Routing (DSR, specified in RFC 4728 [98]) [99]. AODV is solely based on the route discovery process. DSR additionally introduces a route maintenance phase where nodes send route error messages and acknowledgments to inform neighbors about broken and active routes, respectively. The concept of route maintenance was incorporated into the Dynamic MANET On-Demand (DYMO) [43] routing protocol. It is based on the AODV protocol and requires less messages compared to AODV. DYMO was further developed and renamed into AODV version 2 (AODVv2) [156] to replace the AODV protocol.

Hybrid Routing Protocols

Hybrid routing protocols combine properties of proactive and reactive protocols. Usually, the route maintenance to nearby nodes is handled proactively. Routes to far away nodes are then discovered reactively using a route discovery strategy. This combination results in better scalability of hybrid protocols compared to purely proactive or reactive protocols. However, hybrid protocols are more complex in their behavior as well as in their implementation.

The most prominent hybrid routing protocol is the Zone Routing Protocol (ZRP) [78, 79]. This protocol uses local routing zones defining the range (in hops) for each node to maintain its routing table proactively. If a sender node wants to transmit a message to a destination node outside the local routing zone it uses a reactive route discovery scheme. By adjusting

the zone range, the protocol can be adapted to special use cases. ZRP also constitutes the basis for many other hybrid routing protocols [2].

Results

Table 2 summarizes our findings above. Overall, reactive and hybrid routing protocols are the most suitable for routing underlay traffic in harsh environments. Especially, the support for large and dynamic networks is important in such environments. However, the presented protocols support dynamic networks at the expense of responsiveness. Since routes are discovered on demand, long delays can be introduced to network packets if routes are discovered for the first time. Our recommendation is to use the AODVv2 [43, 156] routing protocol for networks with high dynamics (e.g., between moving participants). In a more static environment (e.g., within the communication infrastructure), the B.A.T.M.A.N. [137] or Babel [51] routing protocol should be used.

	Low overhead traffic	Responsiveness	Small routing information	Support large networks	Support high dynamics
Requirements	R.R1	R.R2	R.R3	R.R4	R.R5
Proactive protocols					
DSDV [158]	—	✓	—	—	—
OLSR [55]	(✓)	✓	—	(✓)	—
B.A.T.M.A.N. [137]	(✓)	✓	(✓)	(✓)	—
Babel [51]	(✓)	✓	(✓)	(✓)	—
Reactive protocols					
AODV [157]	(✓)	—	✓	✓	(✓)
DSR [98, 99]	(✓)	(✓)	✓	✓	(✓)
AODVv2 [43, 156]	✓	(✓)	✓	✓	(✓)
Hybrid protocols					
ZRP [78, 79]	(✓)	(✓)	(✓)	✓	(✓)

Table 2: Comparison of presented routing protocols

3.2 Distributed Service Provision

In Chapter 2, we introduced our concept of the peer-to-peer service overlay. It is a peer-to-peer-based system for consuming and providing various services on top of a mobile ad-hoc network. We identified requirements to be fulfilled by service overlays for harsh environments and presented a coarse system architecture. Furthermore, we highlighted the importance of service placement and service discovery in harsh environments. In the present section, we investigate state of the art approaches related to peer-to-peer service overlays.

Our peer-to-peer service overlay is based on mobile services. Hence, we start with the core concept service migration in Section 3.2.1. Afterward, we investigate distributed systems designed for service provision in Section 3.2.2. We also investigate concepts that could, in principle, fulfill the requirements identified in Section 2.2. We evaluate state of the art service placement (Section 3.2.3) and service discovery (Section 3.2.5) approaches for their applicability to harsh environments. Finally, we briefly investigate state of the art approaches to be used as the remaining system components identified in Section 2.3 (Section 3.2.7).

3.2.1 Service Migration

First off, we stressed on the fact that peers in harsh environments are mobile, thus, moving through the geographical region and constantly changing the topology of the underlay network (cf., Chapter 2). We further pointed out that to cover the non-functional requirements performance, costs, and efficiency the service instances have to be migrated during runtime. Migrating applications between computing devices during runtime requires to transfer the application code or executable binary file as well as the current execution state to the new host. The key concept to achieve mobility of service instances during runtime is code mobility.

Code mobility is a rather old concept and considered established in today's research. Fuggetta et al. provide an extensive overview on code mobility in their 1998 survey on mobile code [72]. For instance, Bharat et al. introduced their vision of migratory applications in 1995 [30]. These applications are not tied to one specific user or device. They can be migrated between devices along with context data and the user interface. The main motivation of their work was to allow users to move from one computing environment to another and let their applications travel with them. Today, complete virtual machines including entire operating systems are migrated between servers during runtime. Clark et al. call this concept "live migration of virtual machines" [54].

Mobile Code Paradigms

In essence, there are four mobile code paradigms [72]:

Client-Server: In the classic client-server paradigm, a remote entity (i.e., the server) provides several services to a local entity (i.e., a client). Upon client request, the server uses the service code to calculate a response and sends it to the client.

Remote Evaluation: In the remote evaluation paradigm, the local entity is in possession of the service code and sends it to a remote entity for execution. The remote entity executes the service code, calculates a response, and sends it back to the local entity.

Code on Demand: The code on demand paradigm is similar to the remote evaluation, however, with exchanged roles. Here, the local entity requests the service code from a remote entity to execute it locally.

Mobile Agent: In the mobile agent paradigm, the local entity executes the service code (i.e., the mobile agent) and migrates it to a remote entity during execution. The mobile agent is suspended and transferred to the remote entity where it is resumed again.

The agent carries gathered knowledge and intermediate results onto the remote entity. From there, the mobile agent can be again migrated to the next entity if necessary.

As we focus on continuous service provision in harsh environments, the mobile agent paradigm is the most suited paradigm for our proposed service overlay. Service instances are agents that can be migrated between peers during runtime. They carry their execution state and all gathered information with them. For instance, Zhou et al. proposed an agent-based middleware in 2007 to support application mobility in pervasive environments [206]. Their middleware is called MDAgent. The system has a centralized layout and application migration is used to follow a user between devices similar to the concept of Bharat et al. [30]. However, the centralized layout of both approaches makes these middlewares unusable in harsh environments.

Results

In conclusion, the concept of migratory applications is very well understood. For a real world implementation of our proposed service overlay, however, there are two additional problems to solve: First, users require their service consumption to be smooth. Hence, the provided service should not be interrupted while the service instance is migrated to a new service provider. Suspending the service instance for the migration process is, thus, not a feasible solution. Second, network connections between service consumer and service provider must be reestablished after the service migration. For a service consumer not to notice this process, continuous connections are required as implemented in the MobileSockets Java library by Okoshi et al. [144]. In the next section, we discuss approaches to realize a service provision platform related to our proposed service overlay.

3.2.2 Approaches for Service Provision in Harsh Environments

In Section 2.2, we identified requirements for a system design to provide versatile services in harsh environments. We further identified components and functionality a system for service provision is required to have. In the last section, we discussed the code mobility concept that allows service instances to be migrated between devices of participants. In the present section, we investigate existing approaches and system designs to provide versatile software services in harsh environments. We investigate peer-to-peer-based service platforms and also approaches from grid and cloud computing that could be applicable to our proposed service overlay. For our investigation, we define the following requirements:

R.S1: Self-organizing: A system for service provision in harsh environments must be able to reconfigure itself autonomously upon changes. This autonomous reconfiguration must be triggered especially in case of participants joining and leaving the system. Furthermore, changes to provided services (e.g., service deletion, instantiation, etc.) must be supported by the system.

R.S2: Support mobile participants: Participants are mobile in harsh environments, thus, moving while consuming services. A system for service provision must support this mobility and provide its functionality even if it is exposed to high dynamics.

R.S3: Support service mobility: In Section 2.3, we discussed the need for mobile services in harsh environments. To ensure high quality of provided services, the system must monitor the services and, if necessary, trigger a relocation of respective services (service placement).

R.S4: Consider peer heterogeneity: In Chapter 2, we stressed that participants have different types of devices with different hardware and software resources. Infrastructure devices further add to the diversity of devices. A system for service provision must not only support heterogeneous devices but it also must consider this diversity when making decisions (e.g., relocating services).

R.S5: No Internet dependency: In harsh environments, Internet connectivity cannot be assumed. Due to destruction and/or communication overload, the Internet link is most probably broken. Hence, a system for service provision must provide its functionality even if no Internet connection is available.

Grid Computing

Service instances require computing resources to provide their functionality to accessing service consumers. Grid computing is a paradigm that allows distributed computing and seems applicable to service provision in harsh environments, at first. Traditionally, computational resources of grid computing nodes are dedicated to solve many and/or large computation tasks [68, 69]. The nodes may be either solely dedicated to a computing grid or they participate during absence of the user. Usually, computing grids consist of a few thousands of nodes, use a centralized management structure, and are mainly used for research purposes (e.g., the computing grid of the large hadron collider at CERN in Geneve [113]). Computing grids, thus, do not consider mobility of participants. Furthermore, their centralized management structure and the dependency on Internet connectivity are further criteria disqualifying grids for harsh environments.

Combination of Grid and Peer-to-peer Concepts

In peer-to-peer systems, on the other hand, every peer acts as server and client simultaneously to reach the common goal (e.g., file sharing). Peer-to-peer networks are fully decentralized and self-organized systems which tend to have millions of participants. Redundancy of resources and the large number of participants make peer-to-peer systems robust to node and link failures. Combining both paradigms to receive a large-scale, self-organizing, and robust computing system was a logical step. Between 2003 and 2006, several approaches were proposed that combined the grid and peer-to-peer paradigms [40, 46, 61, 68, 190].

The approaches by Uppuluri et al. [190] and Chakravarti et al. [46], both published in 2005, propose to harness the computing capabilities of personal desktop computers. The computers are interconnected over the Internet forming a self-organizing peer-to-peer network. Computation task are then distributed over the network to all participating nodes in a decentralized fashion. The main goal was to eliminate the centralized management structure for resource allocation and task distribution. Caromel et al. presented a middleware for programming and running applications on grid and peer-to-peer systems in 2006 [40]. Also in 2006, Eom et al. proposed to combine peer-to-peer and grid technology for multiplayer

online gaming [61]. However, none of these approaches considered mobility of participants and all approaches require an always online Internet connection.

Peer-to-peer Service Platforms

Overall, the peer-to-peer paradigm is the best fit for harsh environments due to its self-organizing nature and high scalability. Although the peer-to-peer paradigm does not directly specify the provision of versatile software services, several works propose to create peer-to-peer service provision platforms. Takasugi et al. developed a peer-to-peer middleware that creates an overlay network for seamless service provision [186]. The main motivation of their work was that users need continuous service consumption across different devices while moving. Especially when traveling, users need a small portable device (e.g., a mobile phone) and, therefore, need their service migrated from the PC to the portable device. Although their approach supports mobile services and, to some extent, mobile users, the main focus of their work is to provide seamless services to individual users. Hence, only the devices of that particular user and a small local peer-to-peer network are considered. A complex and highly dynamic environment (i.e., harsh environment) is not generally supported.

Peer-to-peer Service Composition Frameworks

Gu et al. developed a service composition framework in 2004 called SpiderNet [76]. SpiderNet is targeted at peer-to-peer-based service provision and service sharing on an Internet scale. They also promoted the term *peer-to-peer service overlay* to be used for this system type. SpiderNet is based on a reliable service discovery and mainly focuses on proactive failure recovery of services to overcome dynamic changes (i.e., departing or failing peers). Simultaneously, Bradley et al. developed the NEMO (Networked Environment for Media Orchestration) framework [37]. NEMO again focuses on service composition over a peer-to-peer-based network. Neither of these frameworks consider the mobility of participants nor the mobility of services. Therefore, neither SpiderNet nor the NEMO framework are suited for service provision in harsh environments.

Operator-grade Peer-to-peer Service Platforms

A first requirement analysis and architecture proposal related to this thesis was done by Kellerer et al. in 2007 [103]. They developed an operator-grade peer-to-peer service platform architecture targeted at user mobility. The main motivation was to provide a peer-to-peer service platform attractive for Internet service provider. At this time, peer-to-peer systems (e.g., file sharing applications) accounted for large amounts of data traffic and Internet service providers tried to prevent this by limiting peer-to-peer-related traffic. Kellerer et al. identified challenges of the mobile environment that can also be mapped to harsh environments: limited and heterogeneous resources of mobile devices as well as the increased node failure probability. They proposed a two-layer architecture consisting of the lower “core peer-to-peer service”-layer (service discovery, information distribution, bootstrapping, etc.) and the upper “application specific peer-to-peer service”-layer (data management, digital rights management, location-based services, etc.). On top of these layers, applications can be developed. However, their architecture relies on an Internet connectivity and they

do not consider the relocation of service instances to increase service quality. Hence, this system is not applicable to harsh environments.

(S)-Labor-Market Models

Chen et al. proposed a DHT-based peer-to-peer system for service provision in 2008 [49]. The system coordinates service groups to adapt to dynamic service demand. They proposed a labor-market model and a recruiting protocol as the basis for their system controlling the interactions between peers and service groups. Liu et al. further extended on that idea in 2011 [117]. They introduced an S-labor-market model based on the concept of super peers. However, both approaches are tailored at handling rising service demands by selecting additional peers for service provision using a recruiting protocol. They neither consider heterogeneity of peers nor the mobility of participants or services.

Cloud Computing

Another paradigm targeted at computing large computation tasks and at the provision of versatile services is the cloud computing paradigm. Vaquero et al. collected several definitions for cloud computing and merged them into one:

“Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also for an optimum resource utilization. This pool of resources is typically exploited by a pay-per-use model in which guarantees are offered by the Infrastructure Provider by means of customized service-level agreements” [191].

Furthermore, there are three different service models for cloud computing:

Infrastructure as a Service: the cloud provider offers a large set of resources, such as storing and processing capacity, to customers (e.g., Amazon EC2⁵).

Platform as a Service: the cloud provider offers a platform that customers can directly use to run their applications on (e.g., Google’s App Engine⁶).

Software as a Service: the cloud provider offers a single application to thousands of customers, usually through the web browser (e.g., Google Mail⁷).

From the definition above and the three service models, it becomes clear that cloud computing requires a large infrastructure such as a data center that is maintained by a cloud provider and accessible through the Internet. This makes cloud computing not applicable to harsh environments.

⁵ Amazon Elastic Compute Cloud website (visited on April 23, 2014): <http://aws.amazon.com/ec2/>

⁶ Google App Engine website (visited on April 23, 2014): <http://appengine.google.com/>

⁷ Google Mail website (visited on April 23, 2014): <http://mail.google.com/>

Combination of Cloud and Peer-to-peer Concepts

Similar to the grid computing paradigm, several works proposed to bring together the cloud computing concept with the peer-to-peer paradigm. In 2007, Samimi et al. proposed service clouds, a distributed infrastructure designed to facilitate rapid prototyping and deployment of adaptive communication services over the Internet [167]. Instead of using resources offered by a cloud provider, they proposed to share computing resources between users in a peer-to-peer fashion. Graffi et al. investigated a similar application scenario in 2010 [75]. They proposed a distributed mechanism to reliably reserve, monitor, and use peer resources in an unreliable peer-to-peer system. Both approaches concentrate on Internet-scale service provision and do not consider mobility of participants or services.

Offloading

Recently, approaches were proposed to enable mobile devices to harness the cloud for executing resource intensive tasks. In 2011, Chun et al. proposed CloneCloud, a system that automatically and seamlessly offloads parts of the execution of mobile applications from a mobile device into the cloud [52]. It optimizes the performance and energy consumption of the mobile applications by migrating the computation intensive part of the mobile application to the cloud. Another approach to offload resource intensive tasks is called μ Cloud, proposed by March et al. also in 2011 [120]. They focused on the offline usability of mobile applications and proposed to compose mobile applications from self-contained components. These components can then be offloaded to the cloud or onto other (mobile) devices in the vicinity. In 2012, Flores et al. also proposed a framework to offload resource intensive tasks to the cloud [67]. Their *Mobile Cloud Middleware* [66] is designed to handle the interoperability issues resulting from different cloud providers and APIs. All these approaches use an Internet link to the cloud to handle resource intensive tasks and to reduce resource and energy consumption on the mobile device. In harsh environments, a working link to the Internet cannot be assumed.

Cloudlets

In 2009, Satyanarayanan et al. proposed the idea of so-called “Cloudlets” [169]. Cloudlets are nearby resource-rich computing nodes which could be harnessed by mobile devices over high-bandwidth, low-latency, one-hop wireless connections. The idea is that users can harness the computing power in their direct vicinity, e.g., in cafés, bars, stores, etc. This would allow them to migrate their services from the cloud or from the mobile device into such a cloudlet. Interaction with the service would then happen directly over a high bandwidth, low latency connection and without the need for a persistent Internet connection. In 2011, Koukoumidis et al. pointed out that computing capabilities as well as memory capacities of mobile devices will continue to grow rapidly [109]. Therefore, they proposed to migrate the entire cloud service (a web search in their case) onto the mobile device and interact with it locally. They called their approach “Pocket Cloudlets”. The concept of cloudlets as such is very promising, however, a complete system design supporting harsh environments is yet missing.

Results

The results of our investigation are presented in Table 3. Due to the properties of harsh environments, none of the approaches above are fully suited to be used as a service provision platform. All of them have more or less severe shortcomings. The grid computing paradigm as well as the cloud computing paradigm are both unusable in harsh environments. Applying the peer-to-peer concept on both paradigms results in a self-organized network. The respective works focus on the elimination of a centralized management entity. Supporting dynamic environments without Internet connectivity is not intended in neither case.

Peer-to-peer service platforms are self-organized by default due to their peer-to-peer base. However, all service platforms lack support for mobile services. Especially when it comes to increasing the service quality, none of the presented approaches considers the solution of relocating service instances during runtime.

	Self-organizing	Support mobile participants	Support service mobility	Consider peer heterogeneity	No Internet dependency
Requirements	R.S1	R.S2	R.S3	R.S4	R.S5
Grid computing					
Computing grids [68, 69]	—	—	—	—	—
Combined with peer-to-peer paradigm [40, 46, 61, 190]	✓	—	—	(✓)	—
Peer-to-peer service platforms					
Seamless service platform [186]	✓	(✓)	—	✓	(✓)
SpiderNet [76]	✓	—	—	(✓)	—
NEMO framework [37]	✓	—	—	(✓)	—
Peer-to-peer service platform [103]	✓	✓	—	(✓)	—
(S)-labor-market model [49, 117]	✓	—	—	—	(✓)
Cloud computing					
Cloud computing [191]	—	—	—	—	—
Combined with peer-to-peer paradigm [75, 167]	✓	—	—	(✓)	—
CloneCloud [52]	—	(✓)	(✓)	✓	—
μ Cloud [120]	(✓)	(✓)	(✓)	✓	(✓)
Mobile Cloud Middleware [66]	—	(✓)	(✓)	✓	—
Cloudlets [169, 109]	(✓)	(✓)	(✓)	(✓)	—
Our proposed approach					
Peer-to-peer Service Overlay	✓	✓	✓	✓	✓

Table 3: Comparison of presented approaches for service provision in harsh environments

The most promising approaches result from the attempt to access computing clouds from mobile devices with small computing resources. Especially, the idea of cloudlets is very

promising and we believe this approach has to be considered in harsh environments. Although cloudlets in their proposed design require a functioning Internet link, this concept can be extended to be disconnected from the Internet cloud. Combined with a self-organized network, the cloudlet concept can be applicable to harsh environments. Hence, we incorporate this concept into our proposed peer-to-peer service overlay for harsh environments. Furthermore, we assume that WiFi routers can also provide (at least light-weight) services to service consumers.

3.2.3 Service Placement

Service placement is the challenge of finding the “optimal” place (peer) for a given service instance under varying circumstances and considering peer mobility. The “optimal” place for service provision depends on different placement goals and might also depend on the application scenario. In Chapter 2, we identified the improvement of service quality to be our placement goal in harsh environments. Below, we list the three most common goals found in literature in addition to our main placement goal:

1. increase the availability of services,
2. fulfill (contextual) requirements of services,
3. reduce network traffic / reduce the load on the entire network,
4. improve quality of services.

Surveys about service placement by Wittenburg et al. [199] and by Ali et al. [11] categorize service instances into two different types: centralized and distributed services. Centralized services are being executed on a single peer providing their functionality to one or more service consumers. Distributed services consist of multiple identical service instances distributed over several peers. By interacting with each other, the service instances provide their functionality to one or more service consumers. As described by Wittenburg et al. [199], service placement approaches are specialized on one service type, either centralized or distributed services. Many service placement approaches are tailored to the placement of distributed services [70, 114, 130, 174]. In this thesis, we focus on centralized services. For instance, the voice communication service is centralized and works similar to the TeamSpeak⁸ application. All members of a chat group (e.g., a group of fire fighters) are connected to one voice chat service instance. When speaking, the voice packets are sent to the service instance to be processed (i.e., mixed) and forwarded to the other group members. In the present section, we, therefore, investigate service placement approaches for such centralized services.

Increase Availability and Fulfill Contextual Service Requirements

Due to node movement, mobile ad-hoc networks tend to partition and connections to service instances brake down easily. To increase the availability of services in such scenarios, Wang et al. [194] proposed an approach to replicate service instances to guarantee service

⁸ Homepage of the TeamSpeak application (visited on February 14, 2014): <http://www.teamspeak.com>

provision during network partitioning. They try to predict the partitioning upfront and start a new service instance in the network that splits off. Riva et al. [162] proposed a system to fulfill contextual requirements. In their particular case, they wanted to inform car drivers about upcoming traffic jams. The authors assume a car-to-car ad-hoc network on a highway where cars can provide services. While driving, the service instance is migrated one kilometer ahead of the driver and informs him about the current traffic situation.

As introduced in Section 2.3, both goals are already covered by other components in our coarse system architecture. The (contextual) requirements of services are fulfilled by the matching component. It selects appropriate peers as service providers according to their available resources and the requirements of the respective services. The availability of services is usually increased using the replication component (however, replicating services can also reduce the load on the network). We investigate the replication component in Section 3.2.7. In harsh environments, the two most relevant placement goals are the reduction of network traffic and the improvement of service quality.

Reduce Network Load

Most approaches for the service placement problem are focused on reducing the service provision costs according to a fixed cost metric, in most cases, network traffic. The RED-MAN middleware by Bellavista et al. [29] manages, retrieves, and disseminates replicas of data/service components to cooperating nodes in a dense mobile ad-hoc network. To achieve this, one node is elected to become replication manager. This replication manager is considered the service instance to be placed. Using simple heuristics, the replication manager is placed in the center of the network topology to minimize distances to all other nodes. This is achieved by iteratively migrating the service instance one hop towards the node with longest distance from the current service provider. This placement approach does not take service consumers or service quality into account. If multiple service instances are provided in the network, the center node might become overloaded quickly when hosting most of these instances.

MagnetOS by Liu et al. [116] is a distributed operating system for ad-hoc networks. The goal of this system is to simplify programming of ad-hoc networks. This is achieved by letting the entire ad-hoc network appear as a single virtual machine. Applications are automatically partitioned into service components. The system is designed to increase energy efficiency. Service components that communicate with each other are placed close to each other in the network topology. It uses time epochs and at the end of each epoch one of five different heuristics is used to determine the new service provider. Again, this placement approach does not consider quality of service in the placement decision.

Service Placement in Ad-hoc Networks by Wittenburg

In his 2008 survey [199] as well as in his 2010 dissertation [198], Georg Wittenburg provides an overview on service placement approaches reducing service provision costs in ad-hoc networks. Service placement approaches considering other cost metrics than network traffic are not considered. In his dissertation [198], Georg Wittenburg proposed two new service placement approaches. One is used for centralized services, the other for distributed services. They are called Graph Cost / Single Instance (GCSI) and Graph Cost / Multiple Instances (GCMI), respectively. GCSI relies on knowledge about the global network

topology. To gather that information the nodes have to continuously exchange information about their neighborhood. The algorithm calculates the distance between all known node pairs. Then, it iterates over all known nodes and selects that node which will most probably have the lowest service provision costs. In large-scale networks, this approach does not scale well and introduces a large data traffic overhead. Furthermore, the network topology changes continuously in highly dynamic mobile ad-hoc networks. We, therefore, conclude that this approach is not applicable to service placement in harsh environments.

Wittenburg also conducted a simulation based study in his dissertation [198] comparing different approaches for the service placement problem. He pointed out that a rather simple, however, efficient approach was developed by Oikonomou and Stavrakakis in 2006 [141] (in fact, it produces results similar to Wittenburg's GCSI approach). They assume tree-like network topologies and achieve optimal service placement by only using local monitoring schemes and iterative service migration towards the node with highest bandwidth utilization. This results in service placement on a node that utilizes the bandwidth of all its links with similar rate. Later, the authors extended their approach to function in general network topologies [142, 143]. We call this approach *tree placement* as it was originally developed for tree topologies.

Results

Service placement is usually considered to reduce data traffic in the overall network. Usually, this placement goal is favorable because a high load on the network increases network congestion. This might further decrease the quality of provided services and may even lead to an outage of service provision. However, we focus on providing high quality services in harsh environments. In terms of voice communication services, this means that the mouth-to-ear delay between participating parties should not exceed 400 ms [93]. Hence, the mouth-to-ear delay or the service round-trip-times must be considered as the cost or quality metric to base service placement decisions on. None of the approaches above are tailored to provide such functionality in harsh environments. We, therefore, propose our own placement algorithms in Section 5.2.

To evaluate our proposed placement algorithms, we select the tree placement approach by Oikonomou et al. [142] to be our evaluation baseline. As to the investigation by Wittenburg [198], this approach has low data traffic overhead and produces good results in terms of low network traffic. In the next section, we describe this approach in more detail.

3.2.4 Service Placement Evaluation Baseline

In the last section, we evaluated the state of the art service placement approaches. We selected the tree placement by Oikonomou et al. [142] to be our evaluation baseline. In the present section, we describe this approach in more detail and how we implement it for our evaluation (cf., Section 5.2).

For his evaluation of the tree placement [142], Wittenburg implemented a slightly simpler version of this algorithm [198]. In our implementation, we adapt these modifications to achieve the same performance. The pseudo code of this approach is provided in Algorithm 1. At the service provider, the algorithm monitors the incoming and outgoing data traffic on

every connection link to the direct neighbors. If more than 50% of the data traffic is passed over one particular neighbor, the service instance is migrated to that neighbor. This ensures a cost reduction because more than 50% of the service messages travel one hop less in the network topology. In our implementation, we base our decision on the last 100 messages. Furthermore, Wittenburg introduced the parameter `INITMESSAGES`. In his implementation, the algorithm waits for at least `INITMESSAGES` to be monitored at the service instance before a placement decision is made. We set this parameter to 100 messages in our implementation to fill our message queue.

```
// placement algorithm for service instance SI on a service provider
// is triggered upon receiving and sending of messages related to SI
begin
  // get total data traffic related to service instance SI
  TOTALTRAFFIC = MONITORING.GETTOTALDATATRAFFIC(SI);

  for each neighbor P in one-hop neighborhood do
    // get data traffic related to service instance SI
    NEIGHBORTRAFFIC = MONITORING.GETDATATRAFFICFORNEIGHBOR(SI, P);

    if NEIGHBORTRAFFIC > TOTALTRAFFIC / 2 then
      SI.MIGRATETo(P);
```

Algorithm 1: Tree placement by Oikonomou et al. [142]

Figure 8 depicts an example network topology for the tree placement approach. Here, service consumers are labeled with the letter ‘C’ and the service provider is labeled with the letter ‘P’. The links used by service consumers for communicating with the service instance at the service provider are drawn in black. Assuming all service consumers produce the same amount of data traffic, the link to the left service consumer will only account for 25% of the overall traffic at the service instance. The link between node ‘P’ and node ‘N’ accounts for 75% of the traffic, thus, the service instance will be migrated onto node ‘N’. In the next iteration, the service instance is again migrated one hop onto node ‘F’, its final location. At node ‘F’, all links are equally utilized, thus, the service instance will remain there until conditions change.

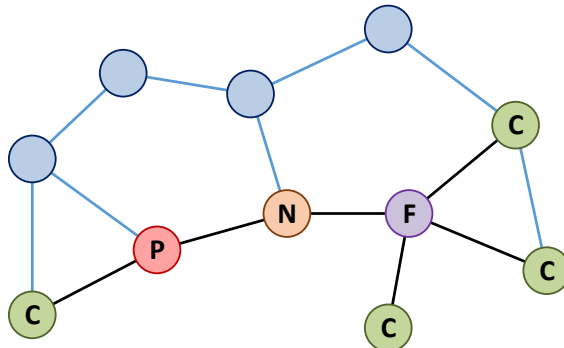


Figure 8: Example network topology for the tree placement approach

3.2.5 Service Discovery

Service discovery is the process that allows network participants to advertise their own services, to query services provided by other participants, and to invoke services [193]. In mobile ad-hoc networks and in service overlays for harsh environments, that discovery process of service objects and service instances is very challenging. The movement of peers and the continuous relocation of service instances caused by service placement introduce high dynamics into the overall system. Hence, we define the following requirements for service discovery:

- R.D1: Scalability:** In harsh environments, the network can consist of several thousands of participants and infrastructure devices. Furthermore, the service overlay will most probably contain multiple service objects and service instances to be managed by the service discovery. Service discovery in harsh environments must, therefore, be able to handle large networks and a large set of services efficiently.
- R.D2: Support mobile participants:** Participants are mobile in harsh environments, thus, moving while consuming, providing, and querying for services. The service discovery must support this mobility and provide its functionality even under high dynamics.
- R.D3: Support service mobility:** In the previous sections, we discussed the need for mobile services in harsh environments. Service placement relocates service instances frequently during runtime. Service discovery is required to discover services even if the service instances are frequently migrated between peers.
- R.D4: Low overhead traffic:** Bandwidth is a valuable resource in harsh environments. Hence, the bandwidth utilization of the service discovery should be low. Especially, the data traffic generated by the service discovery should not congest the network.

In 2008, Ververidis et al. published a detailed survey about service discovery in mobile ad-hoc networks [193]. The different approaches can be classified into directory-based, directory-less, and hybrid solutions. Usually, directory-less approaches use (controlled) flooding to search for services in the entire network. In directory-based discovery approaches, a (de-)centralized directory is used where service objects and service instances are registered at and requested from. Finally, hybrid solutions combine the advantages of structured and unstructured peer-to-peer systems to realize service discovery. Figure 9 presents the classification of approaches for service discovery in mobile ad-hoc networks.

In addition to the classification, service discovery approaches also differ in how they manage the service information (the service description and the information about the service provider). In general, the approaches allow either participants to query for services or service providers to advertise their services. Hybrid solutions are possible as well. The question of what concept to combine with what information management depends on the parameters of the mobile ad-hoc network. As to Mian et al., the main factors influencing the performance of service discovery are the network size and the node mobility [124]. In their 2009 survey about service discovery in mobile ad-hoc networks, they concluded that with higher mobility service directories become less applicable due to frequent topology changes. They further conclude directory-less approaches based on flooding to work best in small mobile ad-hoc networks with high node mobility.

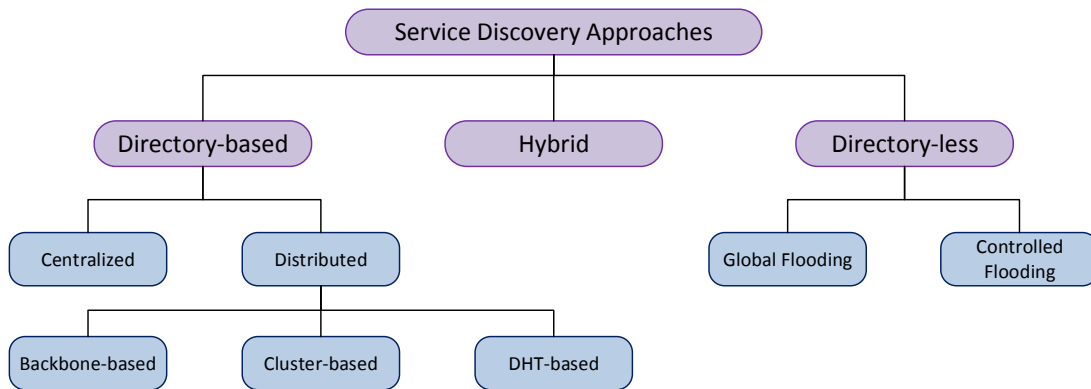


Figure 9: Classification of service discovery approaches by Ververidis et al. [193]

In the present section, we investigate several approaches for service discovery from the three categories of service discovery: directory-based, directory-less, and hybrid approaches. We evaluate them according to the requirements defined above.

Directory-based Approaches

In directory-based approaches, nodes can take on different roles. In addition to the roles of service providers and service consumers, they can also act as directory nodes. Service providers announce their services at the service directory and service consumers solely retrieve information about services from the service directory. The service directory can be either centralized on one (logical) directory node or decentralized on multiple directory nodes. Considering harsh environments, one or several stationary nodes with unlimited power (if available) might be most suitable to host the centralized directory. Such a powerful directory node would be easily accessible for all participants and it would be more robust compared to a mobile node hosting the service directory. The most prominent centralized approach to directory-based service discovery is the Universal Description, Discovery, and Integration (UDDI) [57] known from classical service oriented architectures. However, centralized directories don't scale well and also constitute a single point of failure. Furthermore, stationary nodes with unlimited power cannot be assumed in harsh environments.

Decentralized Approaches

Decentralized service directory approaches are more suitable for harsh environments in terms of scalability. A prominent approach for service discovery with decentralized directories is Jini [14]. In Jini, several nodes are selected to be directory nodes, also called “lookup server”. Lookup servers do not communicate with each other. Service providers have to decide for themselves which lookup servers they want to announce their services at. Lookup servers advertise registered services in a predefined circle around them. But, due to high network dynamics (movement of participants), entries in a single service directory node can become obsolete quickly. Furthermore, potential but distant service consumers might not be able to discover the provided service, although, it is available. The mobility of services is not supported explicitly with this approach.

Backbone-based Approaches

Several directory-based approaches introduce communication schemes between directory nodes. The directory nodes then continuously exchange information about registered services. Kozat et al. build a backbone of directory nodes with the help of a minimum dominating set algorithm [110]. Service providers announce their services at several members of that backbone. If a backbone node cannot respond to a query, it disseminates the query randomly to other backbone nodes. Sailhan et al. use a similar approach but instead of randomly disseminating the query among backbone nodes only the direct neighbors are contacted [165]. This reduces network traffic, however, false negative queries are more likely. In both approaches, frequent exchange of directory information among backbone nodes increases the probability of a positive query. However, these backbone-based approaches also do not support service mobility explicitly. The movement of participants, however, is supported to some extent.

Cluster-based Approaches

Cluster-based approaches, e.g., Service Rings by Klein et al. [107], can be used as an alternative to backbone-based approaches for decentralized directories. This approach groups service providers into clusters (also called “rings”) based on physical distance or similarity of service description. Each ring contains a node, called service access point, which is responsible for service announcements and service queries. It also exchanges information with service access points from other rings. Klein et al. further extended their approach by introducing hierarchical clustering of service providers [108]. Schiele et al. also proposed a cluster-based approach [171]. With their approach, all nodes are grouped into clusters according to their movement pattern. Every Cluster contains a “clusterhead” which stays active permanently and responds to service queries. All other nodes only wake periodically to inform the clusterhead about their availability and their provided services. New clusterheads are voted regularly to balance the load and to reduce energy consumption of clusterheads. Again, service mobility is not targeted with these approaches.

DHT-based Approaches

Another prominent solution to decentralize the directory is an approach known from peer-to-peer systems: a distributed hash table (DHT). Prominent examples for DHTs are Chord by Stoica et al. [185], Pastry by Rowstron et al. [163], and Kademlia by Maymounkov et al. [121]. In DHTs, services are assigned identifiers (hashes) from an identifier space. Nodes are responsible for a subset of this identifier space, thus, storing information and responding to service queries about services from their respective subset. Furthermore, DHTs exist especially designed for the use in mobile ad-hoc networks. For example, the peer-to-peer systems Ekta [159] and MADPastry [204] extend on the Pastry DHT to be used in a mobile ad-hoc network. They try to combine the Pastry DHT with mobile ad-hoc network routing protocols to benefit from a cross-layer approach. However, these approaches introduce a rather large overhead on the system when the network is exposed to high dynamics. Hence, we do not consider them for harsh environments.

Directory-less Approaches

Directory-less approaches have the benefit of reduced overhead compared to directory-based approaches. Since no directory is needed to mediate between service providers and service consumers, the overhead for maintaining the directory is not required. With such approaches, service providers advertise their services in the network and service consumers broadcast their service query through the network. Both processes can happen in parallel. In early approaches to directory-less service discoveries, service providers responded directly to service queries of potential service consumers. Such global flooding schemes are generally not applicable to harsh environments due to the heavy data traffic imposed on the network.

Later, caches were introduced on intermediate nodes to respond more quickly to a received query [131]. The approach by Nidd, called “scheduling and prioritization”, is an early service discovery approach using caches [139]. A node periodically sends information about started services to its direct 1-hop neighborhood. This information includes local as well as remote services on neighbor nodes and has a predefined period of validity. Furthermore, an exponential back-off algorithm manages the intervals between broadcasts depending on the priority of service providers and network changes. Helal et al. proposed an approach called “Konark” which uses multicast to reduce the network load generated by broadcast messages [84]. Service advertisements as well as service queries are sent to fixed multicast groups. This way, the messages are distributed in the entire network but with reduced data traffic.

Controlled Flooding

Flooding the network with broadcast or multicast messages is very costly, thus, more efficient approaches are needed. For instance, the range of advertisements can be reduced to a specific number of hops as in the Group-based Service Discovery (GSD) approach by Chakraborty et al. [45]. This approach uses selective forwarding where a node that cannot respond to a query forwards the query to selected nodes which can respond to the query more likely. An alternative to selective forwarding is “probabilistic forwarding” used by Gao et al. in their protocol RIFCFFP [73]. In this case, nodes forward a query which they cannot respond to with decreasing probability on each hop. Another technique is “intelligent forwarding” used by Nedos et al. [135] where each node monitors its 2-hop neighborhood and calculates a small set of 1-hop neighbors to reach the entire 2-hop neighborhood. Nodes in this set are called “broker nodes” and messages are only forwarded by these broker nodes to reduce the network load. These approaches are supposed to have low overhead traffic and might be most suited for service discovery in harsh environments.

Hybrid Approaches

In hybrid approaches to service discovery, a service provider announces its services at a service directory if it knows about a directory node. If no directory node is available, service providers simply send broadcasts and flood the entire network advertising their services. Service queries are handled the same way. Furthermore, either the service providers or the service directory nodes can respond to queries [193]. Since almost all directory-based

approaches can be extended to such a behavior, we do not investigate hybrid approaches in more detail.

Results

We summarize our results in Table 4. Overall, distributed directory-based approaches increase the load on the network due to the management of the directory. This includes additional message overhead as well as energy consumption of devices. Furthermore, directory-based approaches only support movement of participants to some extent. Movement of participants (and also of services) can out date the service information stored in the directory quickly. This in return leads to false positive query responses.

Directory-less approaches, on the other hand, have usually less traffic overhead (provided caches or controlled flooding are used). In general, changes are propagated quickly through the network. Hence, movement of participants as well as of services are better supported with such approaches. Especially, the controlled flooding approaches seem to be the most suited service discovery approaches for harsh environments.

	Scalability	Support mobile participants	Support service mobility	Low traffic overhead
Requirements	R.D1	R.D2	R.D3	R.D4
Directory-based approaches				
Centralized [57]	–	(✓)	(✓)	✓
Backbone-based [110, 165]	(✓)	(✓)	(✓)	–
Cluster-based [107, 108, 171]	(✓)	(✓)	(✓)	–
DHT-based [159, 204]	(✓)	(✓)	(✓)	–
Directory-less approaches				
Global flooding [84, 131, 139]	–	✓	(✓)	(✓)
Controlled flooding [45, 73, 135]	(✓)	✓	(✓)	✓

Table 4: Comparison of presented service discovery approaches

None of the investigated approaches have an explicit support for service mobility. The reason for this is that the authors do not consider service placement in their system design. We, therefore, investigate how service placement influences the performance of selected service discovery approaches. For this investigation, we select the directory backbone approach by Sailhan et al. [165] to represent directory-based approaches. It has a reduced overhead compared to the approach by Kozat et al. and can also cover cluster-based and DHT-based approaches due to their similarity. As we identified controlled flooding approaches to best fit as service discovery in harsh environments, we further select two approaches from this category. We select the approaches by Chakraborty et al. [45] (GSD) and by Nedos et al. [135] (Service*). Although both approaches are quite similar to each other, they both promise to achieve good performance in terms of service discovery and, at the same time, low costs. We describe the three selected approaches in more detail in the next section.

3.2.6 Selected Service Discovery Candidates

In the last section, we investigated several approaches for service discovery in harsh environments. We selected three candidates to be implemented and investigated for their support for service mobility. One approach is directory-based, the other two approaches are directory-less approaches. Our focus lies on the interplay between service placement and service discovery. Service placement is responsible to relocate service instances during runtime. The service discovery must be able to discover services even if they are relocated frequently. In the present section, we describe the selected candidates for service discovery in more detail and how we implement them for our evaluation (cf., Section 5.3).

Directory Backbones by Sailhan et al. [165]

The main idea of this directory-based approach by Sailhan et al. is to select single directory nodes responsible for a small group of nodes in the vicinity (within \mathcal{H} hops). Service providers send their service advertisements to one or more directory nodes in their vicinity. Directory nodes exchange information about advertised services with each other, thus, forming a backbone of directories. Due to this information exchange, participants can query their local directory nodes for every service in the network.

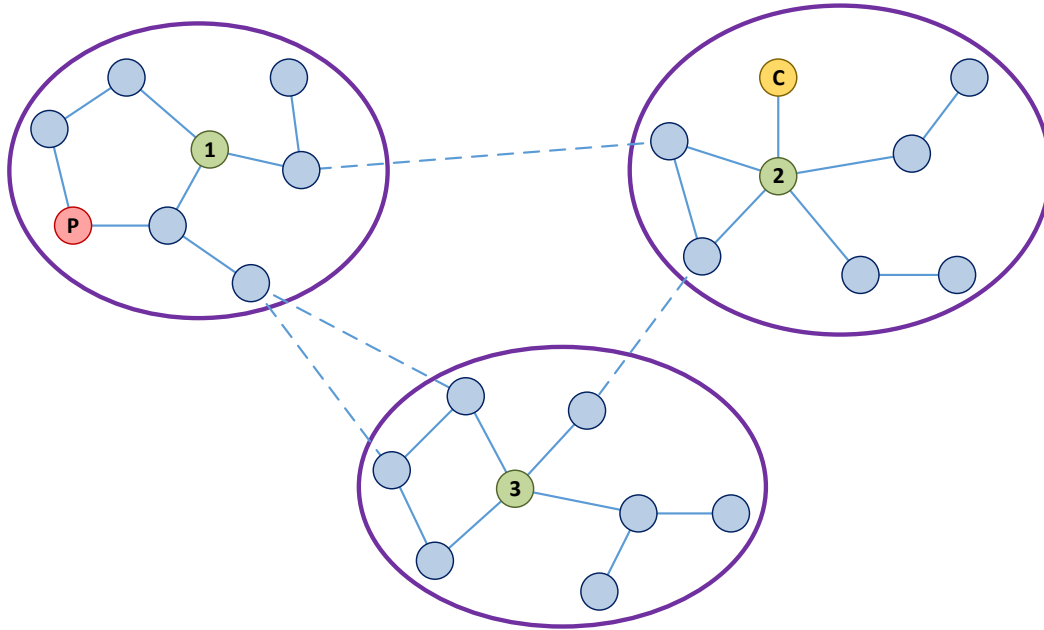


Figure 10: Directory nodes ('1', '2', and '3') forming the backbone of directories

The scheme is illustrated in Figure 10. The service provider node (labeled 'P') registers all services at once at one or more directory nodes in its vicinity (in this case at node '1'). Afterward, the service provider only needs to send periodical keep-alive messages to the directory node(s) to inform the directory that its services are still active and available. The directory nodes '1', '2', and '3' inform each other about new services and periodically exchange keep-alive messages, as well. To discover a service, a potential service consumer

(label 'C') sends a service query to one or more directory nodes in its vicinity (node '3'). If this directory node is aware of the searched service, it responds to the query. If the queried service is unknown to the directory node, it forwards the query to all known directory nodes that then respond to the query. If no directory node is known to the potential service consumer (node 'C'), it becomes a directory node itself and starts a query for other directories. To better adapt to the mobility of participants and the dynamics of the network, directory nodes become ordinary nodes again after a predefined period of time, thus, another node will become directory node. For more details on this approach, the reader is referred to the work by Sailhan et al. [165].

This scheme uses several types of messages which are sent through the network: advertisement of services, query for services, query for directories, and maintenance of directories. Advertisements expire after a predefined period of time, in our implementation after 150 seconds. We set the timeout of service queries to 5 seconds. Also, directory nodes become ordinary nodes after 10 minutes. Furthermore, we set the interval between keep-alive messages to 60 seconds and the directory node advertisement interval to 60 seconds, as well. We set the diameter of directory nodes to $\mathcal{H} = 2$ hops, which is the maximum distance between directories to be aware of each other.

Service* by Nedos et al. [135]

The goal of the directory-less approach called Service* by Nedos et al. [135] is to discover services globally in the entire network with minimum search time and reduced message overhead. To reach this goal a special role for nodes is introduced, the so-called service brokers. Service brokers disseminate service information across the network. They receive advertisements about services from service providers or other service brokers and forward them to nodes and to other service brokers. The basic idea of service brokers is to reach all nodes within the 2-hop neighborhood by sending messages only to a subset of the 1-hop neighborhood.

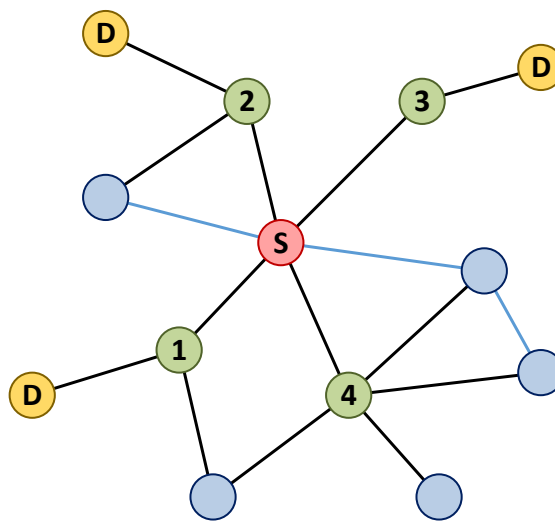


Figure 11: Selection of the service brokers ('1', '2', '3', and '4')

Service brokers are selected using a broker selection algorithm. We briefly describe this algorithm by means of Figure 11. The node labeled with the letter ‘S’ initializes the broker selection algorithm. In the first step of the algorithm, nodes in the 2-hop neighborhood are identified which can only be reached through specific nodes in the 1-hop neighborhood. These nodes are labeled ‘D’ in Figure 11 and can only be reached through nodes ‘1’, ‘2’, and ‘3’. Therefore, nodes ‘1’, ‘2’, and ‘3’ are considered for service brokers. In the second step of the algorithm, all nodes are calculated that can be reached through the current service brokers. If there are still nodes unreachable from the 2-hop neighborhood, that 1-hop neighbor is selected as service broker which can reach most nodes from the 2-hop neighborhood. In this case, node ‘4’ is selected. If all nodes from the 2-hop neighborhood can be reached through the selected service brokers, the algorithm is finished. For more details about the Service* protocol and the broker selection algorithm, the reader is referred to the work by Nedos et al. [135].

In our implementation, every node knows all its direct (one-hop) neighbors. Since the scheme works with expiration of service information, we set the advertisement timeout to 150 seconds and the interval between advertisements to 60 seconds. The broker selection algorithm is triggered every 120 seconds and the broker request timeout is set to 5 seconds.

Group-based Service Discovery (GSD) by Chakraborty et al. [45]

The directory-less approach called Group-based Service Discovery (GSD) by Chakraborty et al. [45] is based on the concepts of peer-to-peer caching of service advertisements and group-based selective forwarding of requests. Every service provider floods his service advertisements periodically in a certain radius (of \mathcal{H}_A hops). The higher the radius, the more nodes can be reached but also more traffic is generated on the network. Service queries, on the other hand, are also flooded in a certain radius (of \mathcal{H}_Q hops) around the querying participant. All nodes that receive an advertisement or a query store this information in their local cache. To reduce message overhead during this process, selective forwarding is used. Using this technique, the query is only forwarded to nodes that might be able to respond to the query more likely. Therefore, a potential service consumer does not need to find the service provider. It is sufficient to reach a node that has the information about the searched service in its local cache. Therefore, the goal of this approach is to find the node where the advertisement diameter and the query diameter intersect.

Figure 12 illustrates how advertisements and queries are sent through the network. The node labeled with the letter ‘C’ represents the participant querying for a service and the node labeled with the letter ‘P’ represents the service provider. The service provider floods its advertisements through the network. At the other side, the potential service consumer also floods its query through the network towards the service provider. Numbers inside nodes in Figure 12 stand for the remaining hops the advertisement or the query are allowed to travel, respectively. The white node in the middle of the figure represents the intersection of both diameters and can respond to the query. For more details about the GSD approach the reader is referred to the works by Chakraborty et al. [44, 45].

In our implementation of this scheme, we set the interval between advertisements to 60 seconds and the timeout of advertisements to 150 seconds. The radius of advertisements is set to $\mathcal{H}_A = 3$ hops. For service queries, we set the timeout to 5 seconds and the radius to $\mathcal{H}_Q = 6$ hops.

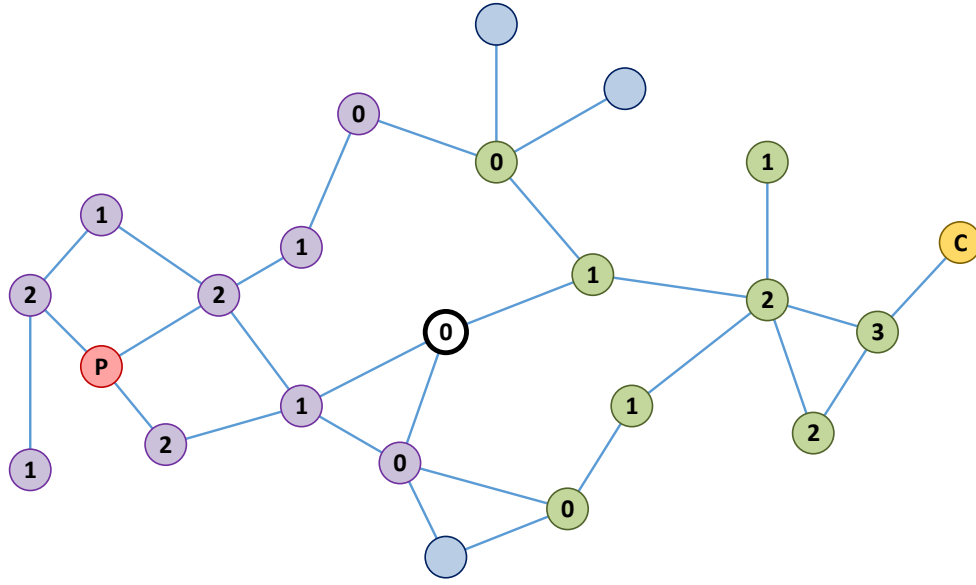


Figure 12: Rendezvous of advertisements (starting at 'P') and queries (starting at 'C')

3.2.7 Other System Components

In Section 2.3, we introduced a coarse system architecture of our proposed peer-to-peer service overlay. We introduced several components providing different functionality of a service overlay. Furthermore, we highlighted the importance of service placement and service discovery in harsh environments. In the previous sections, we investigated related concepts to design service overlays for harsh environments and evaluated state of the art service placement and service discovery approaches. In the present section, we provide a brief overview on the remaining system components: service registry, matching, replication and failover, monitoring. We investigate existing solutions that could be used as these system components to provide the needed functionality.

Service Registry

The service registry provides functionality to add, search, and access services in the service overlay (cf., Section 2.3). In most systems, this functionality is covered by the service discovery as it already provides means to search and access services in the system [37, 76]. Depending on the type of service discovery, adding (i.e., registering) services is possible, too. But as we discussed in Chapter 2, the properties of harsh environments require further components of the service registry. These components are the service placement and the matching components. We investigated service placement and service discovery above and present existing solutions to the matching component in the next section.

Matching

In harsh environments, different types of devices must be handled. Devices vary in terms of their hardware specifications and available resources for service provision. At the same time, services have varying resource requirements for service execution. This leads to a problem: the heterogeneity of devices in terms of hardware and software resources does not allow the execution of any type of application. Especially, resources of peers do not automatically match the requirements of service objects for execution. Peers cannot execute and host service instances requiring other resources than are available. Hence, a mechanism is needed to match resource requirements of service objects and available resource capacities of potential service providers. Such a mechanism is called matching.

In most state of the art matching approaches, this problem is broken down to finding the maximum flow in a bipartite graph [106, 115, 195, 201]. Figure 13 shows a simplified example for a bipartite matching of services and peers. All service objects and all peers are mapped onto the bipartite graph and are represented by one group of nodes, respectively. A service node on the left is connected to every peer node on the right that is in principle capable of providing the necessary resources for hosting the respective service instance. These connections are weighted edges, where weights correspond to the costs of all needed resources (i.e., CPU, memory, etc.). A source node and a sink node are inserted and connected to every service and every peer node, respectively. Finally, a maximum flow from source to sink has to be calculated such that every service instance is placed on one peer at most (splitting a service is not allowed).

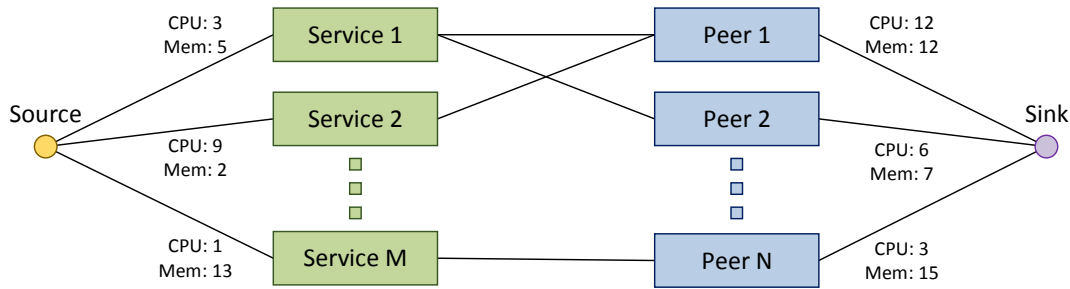


Figure 13: Bipartite matching example

In harsh environments, it is most important that a matching algorithm is distributed and does not require a central instance coordinating the matching process. Wattenhofer presented a fast and fully distributed algorithm for the matching problem [195]. This approach would be a good fit for our peer-to-peer service overlay in harsh environments.

Replication and Failover

Peer-to-peer systems are exposed to high churn rates (peers connecting to and disconnecting from the system). This compromises the availability of objects (e.g., service objects) distributed in the peer-to-peer system. Adding peer and service mobility in harsh environments further increases this problem due to possible connection loss resulting from limited

range of wireless hardware. This high dynamic requires a replication mechanism to automatically replicate popular services, thus, increasing the availability of service objects and also the robustness of the entire system.

A comprehensive survey on different replication techniques is provided by Amjad et al. [12]. In peer-to-peer service overlays the replication of service instances also includes their execution states which have to be synchronized continuously. For example, with passive replication a primary service replicates itself onto a second peer and transmits state updates to synchronize the execution states [200]. If the primary service provider fails, a replicated service instance must take over; this is called failover [23].

The approach by Snoeren et al. groups service providers into so-called support groups to quickly and efficiently take over, if group members fail [177]. Such an approach would be a good fit for peer-to-peer service overlays in harsh environments, provided the support groups are close by in the network topology. In particular, a combination of the placement and the replication component could enhance this approach further. Once a service instance is migrated to another peer nearby, the instance is not shut down on the old peer. If the new service provider fails, the old service provider is selected for the failover process.

Monitoring

In peer-to-peer service overlays, service instances can be subject to (severe) failures or even to misuses from internal or external sources (i.e., cyber-attacks). Thus, the current state of the network and the peers must be *monitored*. As discussed earlier, service placement also requires monitoring information about the current local network state and about the executed service instances to make decisions. Hence, the gathered data must also be available to other system components, e.g., service placement, service discovery. This helps to analyze running service instances and to trigger possible countermeasures in critical situations. Finally, the monitoring component must also oversee the functionality of the other components to ensure high system performance.

Monitoring schemes known from literature mainly differ in the overlay topology they create to exchange monitored information. Monitored information is also called attributes. In literature, meshes [96, 104] and trees [119, 202] constitute the two prominent monitoring topologies for a decentralized environment. In tree-based monitoring schemes, the tree topology is built on top of a well-defined or stable structure of the communication network. The information is only exchanged between peers who are direct children and parents in the virtual tree topology. But, a well-defined or stable network structure cannot be assumed in harsh environments. Hence, tree-based monitoring schemes are not suitable for our peer-to-peer service overlay.

In mesh-based monitoring schemes, attributes are exchanged randomly among participating peers using a mesh topology [62]. One or several direct neighbors are randomly chosen to exchange monitored information. This information exchange within mesh-based monitoring mechanisms is also called gossip-based communication. The communication links are limited to the nearest neighborhood, i.e., physical or overlay neighborhood. Mesh- or gossip-based monitoring schemes are suitable in decentralized environments with no well-defined structure or with a dynamic structure. Hence, these monitoring schemes are the best fit for our peer-to-peer service overlay in harsh environments.

3.3 Simulation Models

In Chapter 2, we argued that simulation models are essential for the reliable performance evaluation of new communication technologies for harsh environments and for disaster recovery, in particular. In the present section, we investigate state of the art simulation models that can be used to reproduce certain aspects of disaster scenarios. Furthermore, we introduce and assess tools for simulation-based investigations of our proposed peer-to-peer service overlay.

For the meaningful investigation of communication systems, it is important to examine a system under similar conditions as they would occur in environments the system was built for. In terms of communication systems, these conditions are represented by the load that is imposed on the system. In disaster relief scenarios, this load is mainly generated by two aspects: the movement of participants and the communication between them. The movement of participants influences the topology of the underlay network, hence, also the performance of the service overlay that uses this underlay network. Therefore, mobility models are necessary to reproduce the movement of all moving and acting entities (rescue workers and civilians) during rescue missions (Section 3.3.1).

The communication and coordination between rescue workers is the second aspect inducing a load on the network. This aspect impairs the bandwidth utilization of the overlay and underlay links in the entire network. It is, therefore, important for the meaningful investigation of the communication system to model the communication of rescue workers realistically (Section 3.3.2). In Section 3.3.3, we present simulation frameworks that already provide some of these simulation models and also provide means to implement missing but necessary models.

3.3.1 Mobility

For the movement simulation of entities in a mobile ad-hoc network scenario, many mobility models have been proposed during the last two decades. Camp et al. provide a comprehensive survey about the commonly used mobility models [39]. As we focus on disaster relief as an example scenario for harsh environments, we concentrate on mobility models that are able to reproduce movement of participants in such scenarios. For our investigation, mobility models are required to generate realistic movement for the different types of participants in disaster relief scenarios, such as civilians, firefighters, ambulances, etc. Hence, we assess the investigated mobility models according to their ability to reproduce realistic movement of rescue workers, vehicles, and civilians in disaster relief scenarios.

In general, mobility models can be categorized into trace-based models and synthetic models. In trace-based models, movement traces are collected from real world systems. These traces are then replayed during simulations to correctly simulate mobility of the network nodes. However, they are bound to the specific scenario where they were gathered. Also, these models are not flexible enough to simulate different scenarios or variations of a single scenario. Unfortunately, no traces are available for disaster scenarios [20].

To simulate movement in disaster relief scenarios, synthetic mobility models can be used. Synthetic models create artificial movement patterns for mobile nodes without replaying recorded real world movement traces. The resulting movement patterns are supposed to

exhibit characteristics of real world traces. To accomplish this goal, the movement and behavior of single nodes must be reproduced realistically. Some of these models use a random component to generate the movement patterns. Other models try to imitate the behavior of the simulated nodes. Therefore, we further categorize the approaches for synthetic mobility models into random mobility models and behavior-based mobility models.

Random Mobility

Although random mobility models are rather simple, they are particularly popular and have been examined extensively in the past [136]. Until today, many random mobility models have been proposed. The four most commonly used mobility models that generate random movement traces for simulated nodes are:

Random Walk Model [77]: In this model, every node picks a direction of the interval $[0, 2\pi)$ at random. Then, the node picks a random velocity and a random time span. The node moves in the chosen direction with the chosen velocity for the chosen time span. Afterward, the node picks a new direction, velocity, and time span and starts the next movement.

Random Direction Model [164]: This mobility model is similar to the random walk model. Here, every node also picks a random direction of the interval $[0, 2\pi)$ and a random velocity. Then, however, the node moves all the way until it reaches the border of the simulation environment. At this point, it picks a new direction and velocity randomly.

Random Waypoint Model [99]: In this model, every node picks a random point in the simulation environment. This point is called a waypoint. The node also picks a random velocity and moves towards the waypoint until it reaches this point. Now, the node again picks a new waypoint and velocity randomly.

Gauss-Markov Mobility Model [39]: This model generates random movement for simulated nodes using a tuning parameter α . Nodes pick a random velocity and direction and start their movement. In fixed time intervals, the nodes change their velocity and direction of movement based on the previous decision (influenced by α). By setting $\alpha = 0$, totally random movement is obtained as velocity and direction are always picked randomly without considering the previous movement. By setting $\alpha = 1$, velocity and direction never change, hence, a linear motion is obtained.

Although these simple random mobility models above are commonly used, they have different unexpected properties and create a behavior that is not usually intended. Yoon et al. [203] showed that in these models the average movement speed of nodes decreases over time. This is because slowly moving nodes need more time to reach their destination than faster nodes.

Random Mobility Models with a Behavior-oriented Component

Alongside those simple mobility models, there also exist models which are not solely based on a random component. Some mobility models try to incorporate a behavior-oriented component into the otherwise random models. These models are:

Pursue Model [168]: Using this mobility model, nodes can be modeled that follow a specific moving target through the simulated environment. For this, every node chooses a random velocity. The movement of an individual node is then defined as:

$$p_{new} = p_{old} + a(p_{target} - p_{old}) + v_{random}$$

The old and new positions of the node are described by p_{old} and p_{new} , respectively. The position of the target is described by p_{target} . The velocity vector v_{random} defines the influence of the random number generator on the node movement. The acceleration $a(x)$ describes how the moving target influences the node movement.

Column Model [168]: Similar to the pursue model, the nodes in this model move on pre-defined lines. These lines move forward toward a random direction from the interval $[0, \pi)$ and for a random distance. This model is suited for certain search activities where nodes move forward forming a front line. For example, if police men should be simulated searching an area for wounded civilians, this model could be used.

Reference Point Group Model [87]: This mobility model arranges nodes into groups. The logical center of each group moves on a randomly chosen path through the simulated environment. All group members move randomly around a predefined reference point which depends on the logical group center. This model can be used in an avalanche rescue scenario, for instance.

For a more detailed overview and a simulation-based comparison of different mobility models the reader is referred to the work of Camp et al. [39].

Behavior-based Mobility

In many scenarios, random-based mobility models do not suffice to reproduce realistic movement patterns. We introduced disaster recovery missions as an example application scenario for harsh environments (cf., Chapter 2). Thus, our goal is to reproduce real world movement and behavior of first responders during rescue missions. In this case, mobility models solely based on random movement cannot reproduce such movement. Especially the different organizations such as police, fire fighters and paramedics have very specific roles when entering a disaster area [136]. Therefore, a mobility model is needed that incorporates these different roles so that the nodes move and act according to their concrete behavior.

Role-based Mobility

Nelson et al. describe a generic event and role-based mobility model [136]. Every node is assigned one role or a set of roles which generate activities for the node to perform on a given event. The entire movement pattern of a node in disaster recovery is then described by a triple (r, e, a) : role r reacts on an event e by performing the activity a . By instantiating the triples with the characteristics for different node types operating in disaster recovery, a movement pattern for the scenario can be generated. This way, it is also possible for a single node to follow different movement patterns during one simulation run. Four different categories of actions are assumed:

Repelling: In disaster scenarios, this role is mainly used for civilians fleeing from the incident. This role can also include a property describing the curiosity of the civilian. The curiosity then defines the possibility that the civilian stops at the periphery of a disaster area, thus, simulating watchers.

Attracting: This role is used for police men and fire fighters moving towards one or more disaster events. It can be combined with further actions once the attracting location is reached.

Oscillating: This role is mainly used to model ambulances moving between the disaster area and the hospitals. The nodes move towards a disaster event and directly after arriving there, they move to a predefined location. They repeat this movement pattern continuously throughout the entire simulation.

Immobile: This role is used to model naturally static objects such as hospitals. But also nodes that become immobile after a disaster event (e.g., injured persons) can be modeled using this role.

Gravity-based Mobility

If several independent disaster areas are simulated, a mobility model based on gravity (proposed by Nelson et al. [136]) can be used to describe how nodes move towards one particular area or away from it. In this model, every disaster area is modeled as a gravity source. The force F a disaster area has on a node can be described by the intensity I as $F = I/d^2$ with d being the distance between the node and the disaster area. The intensity I can also be negative to describe nodes that move away from the disaster area (i.e., fleeing civilians). For a particular node, all force vectors \vec{F} from multiple disaster areas are then combined to the vector sum \vec{F}_{total} . This vector finally describes the node's resulting speed and direction of movement.

Zone-based Mobility

Aschenbruck et al. [16] describe a partitioning of the entire disaster area according to handbooks for first responders in Germany [128, 173]. This partitioning then influences the movement of the rescue workers. The disaster area is divided into four different zones:

Incident site: The center of the disaster area is the incident site. It is the zone where the actual disaster occurs. In this zone, the effects of the disaster have to be combated (e.g., fire). Also, casualties are to be expected and need to be rescued.

Treatment zone: After they have been rescued from the incident, all injured people are brought to the treatment zone. Here, the injured get an extended first aid treatment by medical personnel. Afterward, the patients are prepared for transportation. Usually, the treatment zone is very close to the incident site.

Transport zone: After their first aid treatment in the treatment zone, patients are brought to the transport zone. In this zone, transportation vehicles such as ambulances and rescue helicopters wait to take patients to the hospitals. The transport zone is usually very close to the treatment zone.

Hospital zone: The patients from the disaster area are being transported by transportation vehicles to hospitals for further treatment. The Hospitals are typically further away from the incident site and are not necessarily part of the actual disaster area.

In this model, every node belongs to at least one of the different zones above. Some transportation nodes move between zones, others only move inside one zone, for instance, fire fighters.

Results

We summarize our results in Table 5. The reference point group model and the Gauss-Markov mobility model are the most commonly used random-based mobility models in mobile ad-hoc network simulation studies [16]. Both models reproduce abstract movement of participants. For an abstract representation of rescue worker movement, the reference point group model would be the best choice. For realistic movement representation of participants in a disaster relief scenario, however, none of the above mobility models are sufficient when used standalone. To generate realistic movement patterns in disaster relief scenarios, the zone-based mobility model by Aschenbruck et al. must be considered. It constitutes the basis for movement of rescue workers in disaster relief scenarios. However, it is a rather simple model and civilians are only passive participants in this models. The role-based mobility model can further increase realism if rule sets are defined properly. We, therefore, conclude that only a combination of multiple mobility models will generate realistic movement patterns for all participants in disaster relief scenarios. We recommend to combine the zone-based mobility model with the role-based mobility model to generate realistic movement patterns for rescue workers and their vehicles. Civilians can be modeled with these models, as well. But, integrating the gravity-based mobility model will further increase realism.

Applicable to	Rescue workers	Vehicles	Civilians
Random mobility			
Random walk model [77]	—	—	(✓)
Random direction model [164]	—	—	(✓)
Random waypoint model [99]	—	—	(✓)
Gauss-Markov mobility model [39]	(✓)	—	(✓)
Pursue model [168]	(✓)	—	(✓)
Column model [168]	(✓)	—	—
Reference point group model [87]	✓	—	—
Behavior-based mobility			
Role-based mobility model [136]	✓	✓	✓
Gravity-based mobility model [136]	✓	—	✓
Zone-based mobility model [16]	✓	✓	(✓)

Table 5: Applicability of presented mobility models to participants in disaster relief scenarios

3.3.2 Communication

The second load aspect on the communication system is the communication between the participants. In disaster relief, the rescue workers coordinate themselves via voice communication. These voice calls must be modeled according to voice calls in real disaster recovery missions. Only then realistic simulations can be performed.

The communication has two aspects to be considered: First, it is important to understand who of the rescue workers speaks when and for how long. Basically, the conversations between individual rescue workers or groups of rescue workers must be modeled. Since we assume an IP-based network, the second load factor induced on the communication system is the amount and size of generated voice packets. In voice over Internet protocol (VoIP) technology, the analog voice is digitized and encoded for transmission using a voice codec. We investigate both aspects separately in the following sections.

Conversation Modeling

Conversation modeling is needed to realistically reproduce conversations during simulations. Especially in communications research, the conversations need to be modeled and reproduced during simulations as they would occur in reality. Only then communication systems can be investigated under realistic conditions. Hence, we assess communication models according to this criterion.

Modeling conversations is a rather old field of research. In the late 1960s, Brady analyzed telephone voice conversations [38]. He identified two basic states of a conversation: the ON-state and the OFF-state. The ON-state describes one person talking and the OFF-state describes silence. He discovered that the periods of these states are exponentially distributed. Finally, his work was the basis for the ITU-T standard P.59 which is a commonly accepted model for artificial conversational speech [92]. In this model, the voice channels are modeled by a two-state Markov model where periods in the ON- and OFF-states are exponentially distributed.

More recent studies showed the exponential distribution of the ON- and OFF-states to be inaccurate and proposed lognormal [27] or Weibull distribution [100, 176] of these states. Based on these works, Aschenbruck et al. modeled the conversations between rescue workers in disaster area scenarios [17]. They observed the voice communication of rescue workers during a large catastrophe maneuver in May 2005 in Cologne, Germany in preparation of the World Youth Day 2005 and the FIFA Soccer World Cup 2006. Based on the analysis of this data, they proposed a three-state (semi-)Markov model and evaluated synthetically generated voice traffic against the obtained measurements.

Their three-state model uses again ON- and OFF-states describing whether a rescue worker is talking or not. These two states describe a single conversation and are called “call holding” and “call idle” in their model. The third state “conversation idle” is introduced to separate individual conversations from one another. This is based on their findings that the period of silence between conversations is larger than silence periods during a conversation. They model the conversation idle times and the call holding times using lognormal distributions. The call idle times are modeled using a gamma distribution.

In follow-up studies, Aschenbruck et al. also investigated conversations between first responders by investigating their radio channel during daily work [18, 19]. By comparing that

to the disaster area scenario, they found that the load in catastrophe situations is higher than in daily first response situations. They identified the extremely stressed and extraordinary situation rescue workers find themselves in during disaster relief to be the cause for this behavior.

Results

Aschenbruck et al. revealed that communication patterns in disaster relief missions differ from that in daily first response missions [17, 18, 19]. In disaster relief missions, the voice load is higher than in daily situations. In this thesis, our goal is to simulate the realistic communication of rescue workers during disaster relief missions. The conversation model for disaster areas by Aschenbruck et al. [17] is, therefore, the best fit in our case. It reproduces realistic voice communication of rescue workers in disaster relief scenarios. We use this conversation model in our simulation-based evaluations.

Voice Codecs

As introduced earlier, the underlay network for communication and service provision is based on a wireless ad-hoc network (cf., Section 2). Therefore, the voice of rescue workers must be digitized and transmitted via the packet-based IP network. For this purpose, the VoIP technology can be used [26, 48].

The main challenge of voice transmission is the digitization and compression of recorded voice in real-time. This step is done by the compression/decompression (codec) technology. The codec is an algorithm that converts audio signals into bit streams and vice versa. The main goal of codecs in general is to highly compress the audio signal for transmission and still provide good quality of the decompressed audio signal. To compress and submit voice packets, each codec introduces a delay to the communication. This delay results from the processing time of the algorithm and the splitting into voice packets. The transmission further adds a delay to the communication. As the overall mouth-to-ear delay should not exceed 400 ms [93], the initially introduced delay should be as small as possible. Hence, we assess voice codecs according to both criteria the voice packet size and the delay introduced by the codec.

Most VoIP applications today use codecs that were standardized by the ITU-T. This ensures interoperability across different VoIP applications and devices. The most popular VoIP codecs standardized by the ITU-T are:

G.711: The ITU-T standard G.711 [91] for audio compression was approved in 1988 and is primarily used in digital telephony. It uses “Pulse Code Modulation (PCM) of voice frequencies at 64 Kbit/s”. G.711 processes audio signals in the range of 300–3400 Hz and samples them at the rate of 8 KHz. Using a non-uniform (logarithmic) quantization with 8 bits to represent each sample results in a 64 Kbit/s bit rate. With this codec, usually 10 ms frames of 80 bytes are sent over the network. Alternatively, some devices send 20 ms frames of 160 bytes per frame. It covers two slightly different encoding versions: “A-law” and “ μ -law” encoding. The μ -law encoding is used in North America and Japan, and the A-law encoding is used in Europe and the rest of the world and also on international routes. This codec has no further compression scheme

of the resulting bit stream. Therefore, it requires only low computation complexity and provides very good voice quality. However, it consumes 64 Kbit/s per direction which is rather high compared to other codecs.

G.723.1: The ITU-T standard G.723.1 [94] is a voice codec that splits the voice stream into 30 ms frames. It is also known as “Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 Kbit/s”. As its name already states, G.723.1 can operate at two different bit rates. Using the high rate excitation (MPC-MLQ algorithm), it operates at 6.3 Kbit/s which results in 24 byte frames and greater audio quality. Using the low rate excitation (ACELP algorithm), it operates at 5.3 Kbit/s resulting in 20 byte frames and fair audio quality. G.723.1 has an algorithmic look-ahead of 7.5 ms duration resulting in a total algorithmic delay of 37.5 ms. Therefore, it introduces a relatively high delay and requires moderate computation complexity. Due to its low bandwidth requirement, it is mostly used in Voice over IP (VoIP) applications. However, music or tones (e.g., fax tones) cannot be transmitted reliably with this codec.

G.729: The ITU-T standard G.729 [95, 166] is a voice codec operating with 10 ms frames. It is officially described as “Coding of speech at 8 Kbit/s using code-excited linear prediction speech coding (CS-ACELP)”. It samples voice at a rate of 8 KHz with a 16 bit resolution. Using a compression algorithm, it delivers a stream of 8 Kbit/s. However, it also can operate at bit rates of 6.4 Kbit/s and 11.8 Kbit/s for worse and better speech quality, respectively. G.729 provides good audio quality due to its high sampling resolution and requires relatively low bandwidth. Therefore, it is mostly used in VoIP applications where bandwidth must be conserved, e.g., conference calls. However, it requires high computation complexity. G.729 has also been extended with further features to reduce complexity resulting in slightly lower voice quality (Annex A) and to suppress silence (Annex B).

Results

The bit rates of the presented voice codecs do not consider the overhead introduced by the lower network layer protocols. Usually, voice is transmitted using the Real Time Protocol (RTP) over User Datagram Protocol (UDP) over Internet Protocol (IP). This overhead has to be added per packet to calculate the real bandwidth consumption which is quite high. To transmit G.729 compressed voice in an Ethernet environment using 10 ms frames (10 bytes packet per frame), the following header sizes must be added: Ethernet header (14 bytes), IP header (20 bytes), UDP header (8 bytes), and RTP header (12 bytes). This results in a total of 54 bytes overhead to transmit a 10 byte payload.

The heavy overhead is one reason for the poor and disappointing VoIP performance in IEEE 802.11 WLAN networks [83]. Furthermore, the performance of VoIP in mesh networks is known to degrade with the number of hops [138]. For these reasons, it is important to reduce both the overhead to payload ratio per packet and the number of hops between sender and receiver. In terms of overhead to payload ratio, we recommend to pack four G.729 frames into one packet. This results in a similar delay as with the G.723.1 codec (40 ms) but with better audio quality. To reduce the number of hops, an appropriate placement

algorithm is needed (cf., Section 3.2.3 and Section 5.2). Placing the service instance such that all or most service consumers are reached with low hop counts should result in higher quality voice communication.

3.3.3 Simulation Frameworks

The investigation of our proposed service overlay for harsh environments and disaster relief requires a versatile simulation framework. It must be able to simulate mobility of rescue workers and the communication between them including the underlying ad-hoc mesh network. State of the art simulation frameworks are usually dedicated to simulate either mobility or communication systems. Usually, a mobility simulation framework is used to generate movement traces for all participants of a given scenario. These traces are then used as input for a network simulation framework to simulate communication systems exposed to the simulated movement. Unfortunately, an integrated simulation framework that provides both comprehensive mobility models and comprehensive communication models is yet missing. We, therefore, present state of the art mobility and network simulation frameworks. We assess these frameworks according to the following requirements:

- R.F1: Mobility models for disaster relief:** The simulation framework should provide disaster-related mobility models as identified in the previous section.
- R.F2: Online interface to other simulators:** To generate realistic simulation results, an integrated simulation is necessary. The basis for such simulations is an online interface to another simulator so that multiple simulators can be connected.
- R.F3: IEEE 802.11 WLAN model:** Before, we argued that the underlay network in harsh environments is based on an IEEE 802.11 WiFi mesh network. To realistically simulate communication using such an underlay, a realistic WLAN model is required.
- R.F4: Peer-to-peer overlays:** We base our peer-to-peer service overlay on the peer-to-peer paradigm. A simulation framework providing ready-to-use peer-to-peer overlays and tools tailored to developing new peer-to-peer systems is required for our investigations.

Mobility Simulation Frameworks

For the simulation of mobility, a plethora of mobility simulation frameworks exists. Therefore, we only concentrate on tools especially designed for generating movement traces for research on mobile ad-hoc networks. A typical tool with such features is MobiSim [132]. Using different mobility models, it generates movement traces in XML format that can be used as input for network simulators. The tool CanuMobiSim [178, 182] further allows to consider obstacles within simulations. The framework Important [22] provides different mobility models and metrics to investigate the performance of routing protocols for ad-hoc networks. Their evaluation results show that the performance of different routing protocols strongly depends on the mobility model used to generate movement patterns. However, none of these frameworks generates realistic movement patterns for disaster relief scenarios.

Urban and Vehicular Mobility Frameworks

Furthermore, there exist several simulation frameworks for the simulation of urban mobility [111] and vehicular networks [64, 82]. The goal of these frameworks is to reproduce typical vehicular traffic for the investigation of car-to-car networks. Although, these simulation frameworks and their models reproduce realistic vehicular movement, they are not really suitable for disaster simulation. This has two reasons. First, these frameworks lack mobility models to reproduce movement of rescue workers during their mission. And second, as introduced earlier, the movement of first responder vehicles is mostly limited to the oscillation between two locations.

BonnMotion Framework

Most of the mobility models we introduced in Section 3.3.1 are implemented in the simulation framework BonnMotion [15]. It is a rich Java-based software that generates synthetic movement traces to investigate mobility in different scenarios. Furthermore, it can be used as a tool for the investigation of mobile ad-hoc network characteristics. The generated movement traces can be exported to be used in several supported network simulators. Especially the implementation of the zone-based mobility model for disaster recovery [16] makes BonnMotion a suitable candidate for our investigations. Unfortunately, BonnMotion lacks an online interface for network simulators. Hence, an integrated simulation is not possible with this framework.

First Response Communication Sandbox

Finally, the first response communication sandbox (FRCS) by Bradler et al. [36] is a simulation framework supporting basic disaster and mobility simulation. It was developed at our group at the Technische Universität Darmstadt. The sandbox provides tools for simulating and visualizing disaster relief scenarios using a tile-based world model. Furthermore, the sandbox provides a preliminary interface for network simulators. In principal, this framework can be used for an integrated simulation. However, this framework lacks state of the art mobility models as compared to the BonnMotion framework, for instance.

Network Simulation Frameworks

For the investigation of communication systems, typically network simulators are used. Until today, a huge variety of such simulation tools exists. The network simulators differ in the design of their simulation engine as well as in the supported network types. Some simulators use a time-step-based simulation engine, such as PlanetSim [74]. PlanetSim uses a central step-clock to simulate timing. Therefore, the developer must decide how much time passes during one step of the central clock. Choosing a large time period for these steps results in less accurate results but at the same time allows to simulate large scale networks more efficiently. Vice versa, small time periods result in more accurate simulation results but simulating large scale networks consumes more time to execute simulation runs.

Most state of the art simulators use a discrete-event-based simulation engine [28, 105, 183, 192, 205]. Here, the simulated network nodes create and enqueue events for specific points in time. These events are processed by a scheduler chronologically without the need for a central clock. The currently processed event determines the current simulation time. Because the developer does not need to define time steps using such a simulation engine

design, one error source is eliminated. Hence, we concentrate on simulation frameworks based on a discrete-event simulation engine.

Low-level Network Simulation Frameworks

One of the first and most prominent network simulators is the Network Simulator version 2 (ns-2) [122]. It has become virtually the standard for network simulation [196]. Unfortunately, ns-2 has some significant shortcomings in the modeling details of the IEEE 802.11 modules. The successor ns-3 [85] is widely used today and will most probably replace ns-2 in the near future. The main goal of the ns-3 project is the improvement of simulation performance compared to ns-2. Both simulators are not compatible with each other requiring the porting of ns-2 models into the ns-3 framework. Unfortunately, both simulators are tailored to simulate lower network layers in great detail. Hence, simulating large scale peer-to-peer systems is not the main focus of these frameworks. Also, both frameworks provide only few peer-to-peer overlays and moderate support for implementing new peer-to-peer systems.

Peer-to-peer Overlay Simulation Frameworks

Simulators, such as PlanetSim [74], OMNeT++ [192] and PeerfactSim.KOM [183], provide means to simulate large scale peer-to-peer overlay networks more efficiently [134]. Especially for the design and investigation of new peer-to-peer systems, such frameworks provide many state of the art peer-to-peer overlays, such as Chord [185], Kademlia [121], and Pastry [163]. These overlays can be used as baselines in evaluations. The efficiency, however, results from an abstraction of the lower network layers. As our proposed service overlay for harsh environments is based on a wireless ad-hoc network, a detailed and correct model of the IEEE 802.11 WLAN standard is required. The most promising candidate for our evaluations is the PeerfactSim.KOM framework. The developers have ported the realistic WLAN model based on the IEEE 802.11g standard specifications [90] from ns-3 into PeerfactSim.KOM to facilitate the simulation of mobile peer-to-peer and ad-hoc networks [184]. This enables us to investigate our proposed service overlay under realistic wireless network conditions.

Results

We present our results in Table 6. In general, simulation frameworks providing online interfaces for other simulators are not particularly common. It is not possible to simulate sophisticated behavior-based mobility and communication models together in one integrated simulation framework, yet. The only framework providing a preliminary interface to other simulators is the first response communication sandbox by Bradler et al. [36]. However, the mobility models provided in that framework are rather simple and cannot keep up with the mobility models implemented in the BonnMotion [15] framework.

Network simulation frameworks, on the other hand, generally lack mobility models for disaster relief scenarios. They concentrate on the simulation of network-related issues and only provide very basic and often random-based mobility models. The only simulation frameworks providing a realistic IEEE 802.11 WLAN model are the ns-3 [85] and the PeerfactSim.KOM [183] simulators. The ns-3 simulator, however, lacks a rich portfolio of

	Mobility models for disaster relief	Online interface to other simulators	IEEE 802.11 WLAN model	Peer-to-peer overlays
Requirements	R.F1	R.F2	R.F3	R.F4
Mobility simulation frameworks				
MobiSim [132]	(✓)	—	—	—
CanuMobiSim [178, 182]	(✓)	—	—	—
Important [22]	(✓)	—	—	—
SUMO [111]	—	—	—	—
VanetMobiSim [64, 82]	—	—	—	—
BonnMotion [15]	✓	—	—	—
FRCS [36]	(✓)	✓	—	—
Network simulation frameworks				
ns-2 [122]	—	—	(✓)	(✓)
ns-3 [85]	—	—	✓	(✓)
PlanetSim [74]	—	—	—	✓
OMNeT++ [192]	—	—	(✓)	✓
PeerfactSim.KOM [183]	—	—	✓	✓

Table 6: Evaluation of presented simulation frameworks

peer-to-peer overlays and comprehensive tools to implement peer-to-peer systems. The best fit for our investigations is, therefore, the PeerfactSim.KOM simulator.

3.4 Summary

In this chapter, we investigated related work for the three major fields of this thesis: peer-to-peer-based service provision, harsh environments (the communication infrastructure in particular), and simulation models. First, we evaluated communication infrastructures for harsh environments and corresponding routing protocols. Our investigation of communication infrastructures revealed that a decentralized communication infrastructure based on the WiFi standard would be the best fit for disaster relief. It outperforms existing standards in terms of throughput, openness and failure resilience. Pre-deployed infrastructures are readily available in disaster situations and can be further extended with roll-out infrastructure. Unfortunately, pre-deployed infrastructures are not widely available. Only few wireless mesh networking projects exist and only in big cities. A more omnipresent infrastructure is yet missing. In terms of routing protocols, we identified reactive protocols to provide higher performance for highly dynamic mobile ad-hoc networks. AODVv2 [43, 156] is the most advanced reactive routing protocol and the best candidate for harsh environments as of now.

Second, we evaluated systems for distributed service provision. We presented the code mobility concept and the mobile agent paradigm which constitute the basis for service mobility in our peer-to-peer service overlay concept. We evaluated general concepts for peer-to-peer-based service provision as well as concepts from grid and cloud computing to be used in harsh environments. Although no concept was able to fulfill the requirements for service provision in harsh environments, the cloudlet concept [109, 169] is the most promising candidate to be considered in our system design. Cloudlets are nearby resource-rich computing nodes (e.g., PCs connected to wireless routers in cafés or bars) that could be used for off-loading complex computing tasks from the mobile device. We further investigated service placement and service discovery. In terms of service placement, most approaches relocate the service instance to reduce network traffic. None of the investigated approaches consider quality of the provided service as metric to relocate the service instance. Therefore, we identified the best performing service placement approach for traffic reduction to be used as our evaluation baseline. This approach is called tree placement [142]. In terms of service discovery, most approaches were designed and evaluated to provide good performance under peer mobility. Service mobility was not considered in most cases. We, therefore, selected three discovery approaches to be evaluated for their performance in combination with service placement: Directory Backbones [165], Service* [135], and GSD [45]. Finally, we briefly investigated the remaining system components and identified suitable approaches to be used in a real world implementation of service overlays.

Third, we investigated simulation models and tools for harsh environments. We investigated mobility and communication models to be used as load factors for simulation-based evaluations. To reproduce realistic movement of all participants in harsh environments (including civilians), a more sophisticated mobility model is required. We found that a combination of zone-based [16], role-based [136], and gravity-based [136] mobility models would produce the most realistic movement patterns for simulations of harsh environments and of disaster relief scenarios in particular. Furthermore, we identified the conversation model for disaster areas [17] and a slightly modified version of the G.729 audio codec [95] to realistically generate communication load for disaster recovery workers. Finally, we evaluated mobility and network simulation frameworks for their suitability as evaluation tools for harsh environments. Unfortunately, an integrated framework that provides versatile mobility models for disaster relief and also allows the realistic simulation of WLAN networks is yet missing. However, we identified the BonnMotion [15] framework to provide suitable disaster-related mobility models and the first response communication sandbox [36] to provide an online interface to connect a network simulation framework. For an integrated disaster simulation we further identified PeerfactSim.KOM [183] to provide the required WLAN model and tools for the development of peer-to-peer systems.

4 Communication Infrastructure

To provide versatile software services in harsh environments, a resilient communication system is required. In harsh environments, however, the communication infrastructure is often destroyed or overloaded (cf., Chapter 2). Furthermore, today's standards exhibit single points of failures and can only deliver low data throughput (cf., Section 3.1). We, therefore, propose a solution based on decentralized communication schemes on top of the IEEE 802.11 WLAN standard. The contributions in the present chapter are threefold:

1. We propose our concept of the *CityMesh*. It is a highly distributed emergency mesh network built from private and public wireless routers in inhabited areas that would be used for disaster relief communication.
2. We present our methodology to evaluate the feasibility of such CityMesh networks by collecting and analyzing data about wireless routers in cities. We further describe a method to estimate real router locations from measured data sets.
3. We evaluate the feasibility of CityMesh networks in several cities and show the severe improvement of network resilience if private routers are used in CityMesh networks.

The contributions in this chapter were published in the proceedings of the international Information Systems for Crisis Response Management (ISCRAM) conference in 2011 [152] and in 2012 [150]. Furthermore, the contributions were published in the International Journal of Mobile Network Design and Innovation (IJMNDI) in 2012 [151] and in the International Journal of Information Systems for Crisis Response Management (IJISCRAM) in 2012 [153]. In addition, our proposed approach attracted international media attention in the second half of 2012. Articles were published on several news websites, such as Ars Technica⁹, Deutsche Welle¹⁰, and Heise Online¹¹.

The remainder of this chapter is organized as follows: In Section 4.1, we present our CityMesh concept in detail and how it could be implemented in the future. We present our methodology for evaluating the feasibility of CityMesh networks in Section 4.2. In Section 4.3, we evaluate the feasibility of CityMesh networks in five selected cities. Afterward, we discuss realization aspects in Section 4.4 and summarize the chapter in Section 4.5.

4.1 Privatized Pre-deployed Infrastructure

In Section 3.1, we investigated several approaches to deploy a communication infrastructure in harsh environments and, in particular, in disaster-affected regions. The TETRA standard,

⁹ News article on Ars Technica (visited on August 23, 2012):

<http://arstechnica.com/tech-policy/2012/08/>

would-you-give-the-government-remote-control-over-your-router/

¹⁰ News article on Deutsche Welle (visited on August 23, 2012):

<http://www.dw.de/the-untapped-potential-of-wifi-in-emergencies/a-16183453>

¹¹ News article on Heise Online (visited on August 30, 2012):

<http://www.heise.de/netze/meldung/WLAN-Mesh-als-Notfunknetz-1676555.html>

used in most European countries today, relies on a backbone-based infrastructure that is based on a cell-type deployment (cf., Figure 14a). The standard provides relatively low bandwidth for data transmission and cell towers constitute a single point of failure. To overcome these drawbacks, decentralized communication systems have been proposed based on the IEEE 802.11 WLAN standard and the peer-to-peer paradigm (cf., Section 3.1.3). Rolling out additional infrastructure (WiFi mesh routers) at the beginning of rescue missions can further increase the resilience of such communication systems (cf., Section 3.1.2). Unfortunately, this comes at the expense of deployment time, additional hardware costs, and maintenance overhead. We believe that in many cities today, enough resources exist to provide a (privatized) pre-deployed infrastructure for decentralized disaster relief communication.

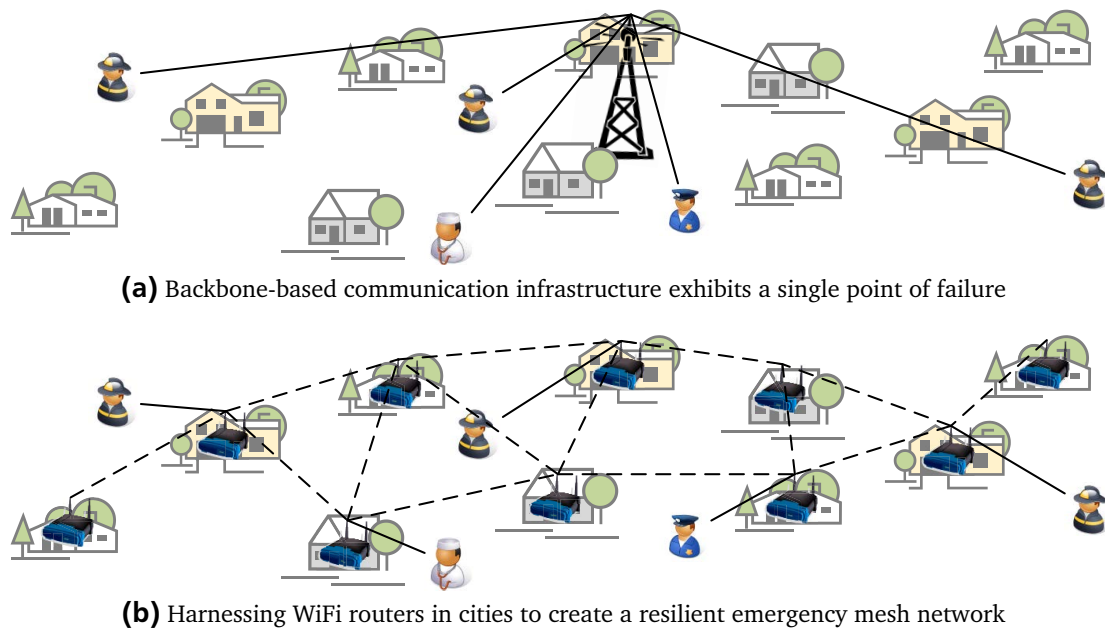


Figure 14: General concept of 'CityMesh'

4.1.1 General Concept 'CityMesh'

With the increasing popularity of smartphones, tablets, notebooks, and other devices capable of mobile Internet access, the number of wireless routers in public venues increases. Internet service providers consolidate with shop owners to provide wireless Internet access in stores, restaurants, and bars. Usually, such routers have no encryption enabled to provide easy access to customers. Especially in city centers, these WiFi equipped public locations occur in high density. We call such routers *public routers*.

The amount of private households equipped with an Internet connection is heavily dependent on the respective country. In economic countries such as Sweden, Singapore, and Denmark more than 90% of the citizens have access to the Internet from their homes according to a study conducted in 2011 [129]. In the United States 80% and in Germany 77% of citizens have access to the Internet from their homes. These numbers are still increasing. Many of those households use wireless routers to effortlessly connect multiple devices to the

Internet, called *private routers*. Especially in apartment buildings, this may lead to a multitude of available wireless routers. This fact can be perceived when searching for available wireless networks in the city center of Darmstadt, for instance. A single WiFi scan in the city center reveals around 20 individual wireless routers to which a connection would be possible (provided the security key is known).

Yet, all this communication capacity in urban areas remains unusable in case of a disaster. We believe these routers should be interconnected to a city-wide wireless mesh network (cf., Figure 14b). We call this network the *CityMesh*. This network could then be used to assist first response communication providing high resilience against individual node failures, high bandwidth, large area network coverage, and almost instant ready-to-use availability. Since the router density roughly increases with the population density, this technology would be most useful where a disaster affects many people, i.e., in urban areas and city centers. In sparsely populated areas, additional roll-out infrastructure could be deployed.

4.1.2 Emergency Switch for Wireless Routers

We, therefore, propose an emergency switch for wireless routers. The emergency switch is meant to transition the router into an emergency mode. All routers in emergency mode would interconnect with one another, thus, forming the emergency mesh network and providing a communication infrastructure for disaster relief communication. This emergency network would also coexist with the usual home network. Technically, this goal could be easily accomplished as it is already today possible to install a home network and a guest network in parallel to grant Internet access to visitors. Such functionality is included in the AVM Fritz!Box 7390 router¹², for instance. We, therefore, believe this emergency switch could be realized with a firmware update for wireless routers capable of such a guest functionality.

Abuse of the emergency switch and the emergency network must be avoided at all cost. Therefore, security protocols for the emergency switch as well as for the emergency network are needed. The emergency switch should only be accessible by the government or another authorized institution. The emergency network must be isolated from the citizen's home network to protect people's privacy and also to protect the emergency network. Legal aspects must be considered and could be solved by identifying individual first responders similar to the concepts in the Freifunk community network (cf., Section 3.1.3). However, the security issues associated with such an emergency switch are not in focus of this thesis.

Finally, there also exist computational resources in the CityMesh network that could be used for service provision. The routers themselves have small computational resources similar to the mobile devices carried by first responders. If these resources were accessible, the routers could provide services in the service overlay, as well. Furthermore, if stationary computers connected to the routers would be accessible through the emergency network, the entire service provisioning system would benefit from large computational resources. Incorporating these resources into the service overlay could greatly support disaster recovery missions. By outsourcing heavy computational tasks to emergency network nodes, first responders would extend battery life of their own mobile devices. This is analog to the concept of cloudlets [109, 169] (cf., Section 3.2.2).

¹² Product website and technical data of the AVM Fritz!Box 7390 router (visited on January 24, 2014): <http://en.avm.de/products/fritzbox/fritzbox-7390/technical-data/>

4.2 Methodology to Investigate the Feasibility of CityMesh Networks

In the previous section, we presented our concept of CityMesh networks and the emergency switch for wireless routers in urban areas. Our goal is to evaluate the feasibility and the resilience of such CityMesh networks. Hence, we describe our methodology to theoretically analyze potential CityMesh networks in the present section. In particular, we aim at answering the following questions:

- What are differences between CityMesh networks and similar random networks?
- Are public routers in cities sufficient to create resilient CityMesh networks?
- What are graph-theoretical properties of CityMesh networks?
- How resilient are CityMesh networks against individual node failures?

To answer these questions, we proceed along three steps. These steps are also summarized in Figure 15. First, we collect data about the locations of wireless routers in selected cities. Second, we construct CityMesh networks from the collected data set. Third, we analyze the constructed CityMesh networks for their graph-theoretical properties and compare the results.

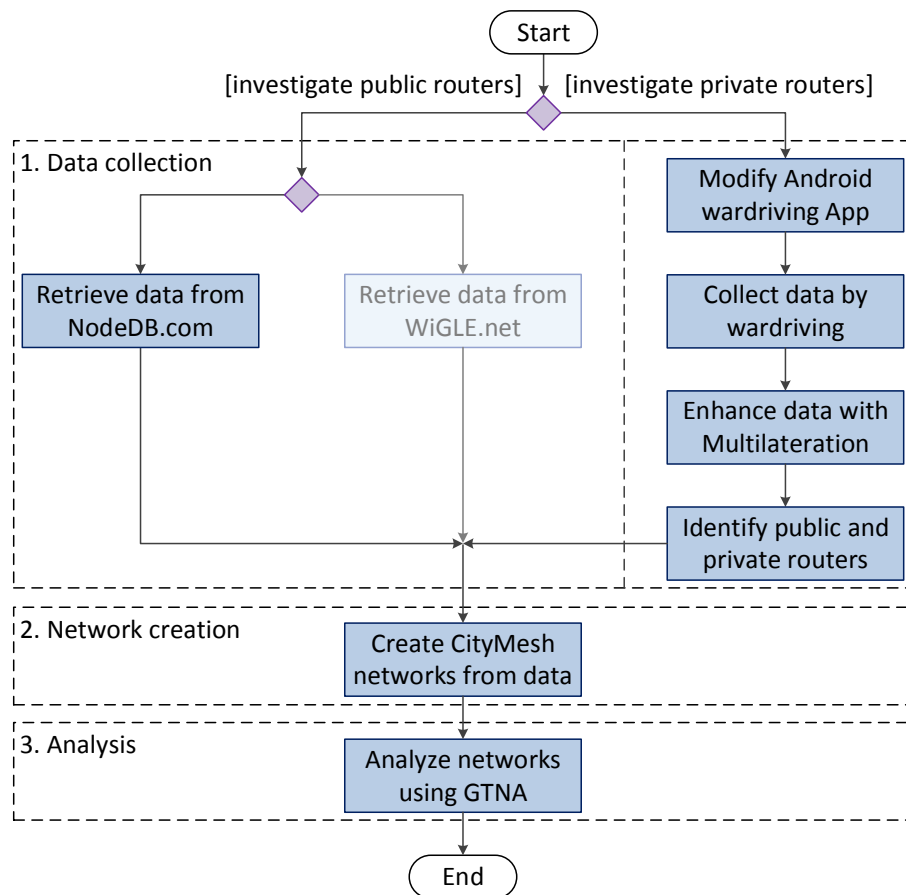


Figure 15: Methodology to evaluate the feasibility of CityMesh networks

As shown in Figure 15, we use different methods to collect data about wireless routers. To obtain data about public routers, we use online data sources provided by public projects. In particular, we use data from the NodeDB project for this task. To investigate private routers, we must be able to identify private and public routers from collected data sets. This requires detailed information about individual wireless routers. We, therefore, collect our own data set in our hometown of Darmstadt using the wardriving technique. Below, we describe both data collection methods, the network construction, and the process of analyzing the created networks.

4.2.1 Data Collection of Public Routers from Online Sources

Online databases can be used to collect information about public wireless routers in a specific area. However, the exact locations of all wireless routers in a specific area are not stored in any database. This would require owners of wireless routers to measure and store the exact location of their routers in a public database. Internet service providers have private information about their customers including the postal address but they are not allowed to share this information. Nevertheless, there are several projects providing approximate data about wireless routers and corresponding locations everywhere in the world. These projects can be categorized as follows:

- i) projects based on voluntary participation of owners of wireless routers, and
- ii) projects based on wardriving and gathering information with crowd sourcing¹³.

In the following, we present one representative from each category. We present the NodeDB project representing the category of voluntary participation of router owners. And we present the WiGLE project representing the category of wardriving-based data gathering.

NodeDB Project

The *NodeDB* project¹⁴ or *The Wireless Database Project* started as an attempt to create a map of wireless Internet access points in Sydney, Australia. It quickly evolved to a world wide mapping service for wireless access points. Using the NodeDB website, people can add and retrieve information about wireless routers into and from a global database, respectively. The information stored in this database includes location, SSID, description, status, internet connectivity, and whether the router is for commercial use or not. Furthermore, the routers are also visualized on a graphical map.

Since NodeDB is an open database and contains data gathered on a voluntary basis, the completeness cannot be guaranteed. In fact, we checked the database for data about our hometown Darmstadt but the result set was empty. Also, the Freifunk community network in Lübeck – one of Germany’s biggest Freifunk networks – is not covered by NodeDB. The Freifunk networks in Berlin, Hamburg, and Leipzig, however, are covered by the project’s database (at least partially).

¹³ Crowd sourcing is a concept of receiving contributions from a large group of people (e.g., from an online community) for needed services or content.

¹⁴ Website of the NodeDB project (visited on May 7, 2014): <http://www.nodedb.com>

WiGLE Project

The *WiGLE* project¹⁵ or the *Wireless Geographic Logging Engine* has the goal to create maps of wireless networks. The project collects world-wide data about wireless networks through crowd sourcing into a central database. While wardriving, community members gather data about wireless routers and upload it to the database. The project supports outputs from various wardriving applications. Users can also add a wireless network manually using an online form. Besides wireless routers, also information about cellular networks is gathered and stored in the database.

Due to crowd sourcing, the WiGLE database is continuously updated and extended. At the time of writing this thesis, the database contained 129,630,058 individual WiFi routers (as of April 6, 2014). The information stored in the database includes location, SSID, BSSID (MAC address), channel, encryption, and signal strength. All data is visualized on a graphical map on the WiGLE website. Furthermore, the project provides several open source applications to access the database and to visualize the data.

4.2.2 Data Collection of Private Routers with Wardriving

To collect detailed information about private wireless routers in a city, we use wardriving. Wardriving is the act of moving around (e.g., by car) and searching for WiFi networks using a WiFi-equipped mobile device. The term originates from wardialing, a method popularized by and named after the movie *WarGames* (1983). Wardialing is a technique to automatically dial a sequence of phone numbers in search of computer modems. Similar to that, wardriving was commonly used to search for unencrypted WiFi networks to get free access to the Internet or to connected computers in the respective home networks. Today, wardriving is mainly used to collect data about wireless networks together with GPS coordinates to be uploaded to online databases.

The main problem with the data provided by projects such as WiGLE is that most of the data points are located on main roads or even highways instead of real router locations. This is because no real estimation is done while wardriving. Wardriving apps measure continuously for wireless network and, thereby, overwrite existing data points if new measurements have higher signal strength values. After the user has uploaded his data set to the database, the WiGLE project tries to enhance the data with post processing methods. This method calculates an average of the collected latitude and longitude values using the measured signal strength (squared) as a weight. This is based on the assumption that signal strength changes at the inverse square of the distance.

Our goal is to improve the data quality by calculating the real locations of wireless routers using lateration methods on the gathered data. Lateration relies on a large amount of collected measurements when doing wardriving and, thus, requires the device to store every measurement made. The wardriving device must record and store multiple data points with GPS locations and different signal strength values for unique wireless routers. For this purpose, we use Android-based mobile devices because they have wireless network adapters and GPS sensors integrated. Furthermore, many wardriving apps already exist for the Android platform some of which also provide open source licenses.

¹⁵ Website of the WiGLE project (visited on May 7, 2014): <http://www.wigle.net>

For our measurements, we use the open source app *wardrive*¹⁶. In the original code, the app scans for wireless networks in the vicinity every three seconds and stores the current time, the current GPS position, and readings from the wireless networks adapter into a local database. Existing database entries to a specific wireless network are replaced with new readings if the GPS coordinates differ more than a specific value and the signal strength of the new data point is higher than the old one. Therefore, only one data point is stored for every wireless router. We modified the application code to continuously log the information into the database, thus, creating multiple entries for individual routers. These data points can then be used to estimate the real location of individual routers using the lateration method.

Measurement in the City Center of Darmstadt

We used our modified version of the wardrive app on a Motorola Xoom Tablet and walked in slow walking speed through the predefined area of our hometown as depicted in Figure 16. This area includes the city center and is roughly $467500m^2$ in size. It is determined by the streets *Hügelstraße*, *Kirchstraße*, *Holzstraße*, *Zeughausstraße*, *Bleichstraße*, *Kasinostraße*, and *Neckarstraße*.



Figure 16: Area of the wardriving measurement

During wardriving, we collected 32983 individual data points. In these measurements, we identified 1971 unique BSSIDs (unique wireless routers). Out of those 1971 routers, 212 routers had no encryption at all (public routers). Of the remaining encrypted routers, 129 were encrypted using outdated WEP encryption, 240 and 421 used WPA and WPA2 encryption, respectively. We also found that 951 routers provided both WPA and WPA2 encryption in parallel. 18 routers had an unknown encryption or were most probably operated by people surfing in cafés with a laptop or tablet using their mobile phone as an ad-hoc access point for WiFi-tethering.

¹⁶ Project website of the wardrive Android app (visited on May 3, 2014): <https://github.com/raffaeleragni/android-wardrive>

In general, lateration is used to estimate locations of wireless networks based on measurements that consist of a location and a measured signal strength of the network. With trilateration [170], exactly three data points are needed for the calculation. But, our modified wardrive app collects a huge amount of data. We, therefore, use the multilateration approach where all measured points to one wireless router are used for the calculation to improve the estimation. The method is depicted in Figure 17 and starts by choosing two arbitrary measurement points P_t and P_u with intersecting circles of the radii r_t and r_u . The radii r_t and r_u are calculated from the measured signal strength values s_t and s_u , respectively. A signal strength s is converted into a distance r by using a formula extracted from the findings of Faria [63] and Rappaport [160]:

$$r = \frac{1}{3} \left(\frac{10(-s - 10)}{35 - 1} \right)$$

We calculate the points in which the circles intersect and denote them $Q_{t,u,1}$ and $Q_{t,u,2}$. Then, a second pair of arbitrary points P_v and P_w are picked with intersecting circles of the radii r_v and r_w , respectively. For these two points, we also calculate the points in which the circles intersect and denote them $Q_{v,w,1}$ and $Q_{v,w,2}$. Now, we pick those two intersection points that are closest to each other (i.e., $Q_{t,u,2}$ and $Q_{v,w,1}$ in Figure 17a) and calculate the point in between. This point is called the centroid.

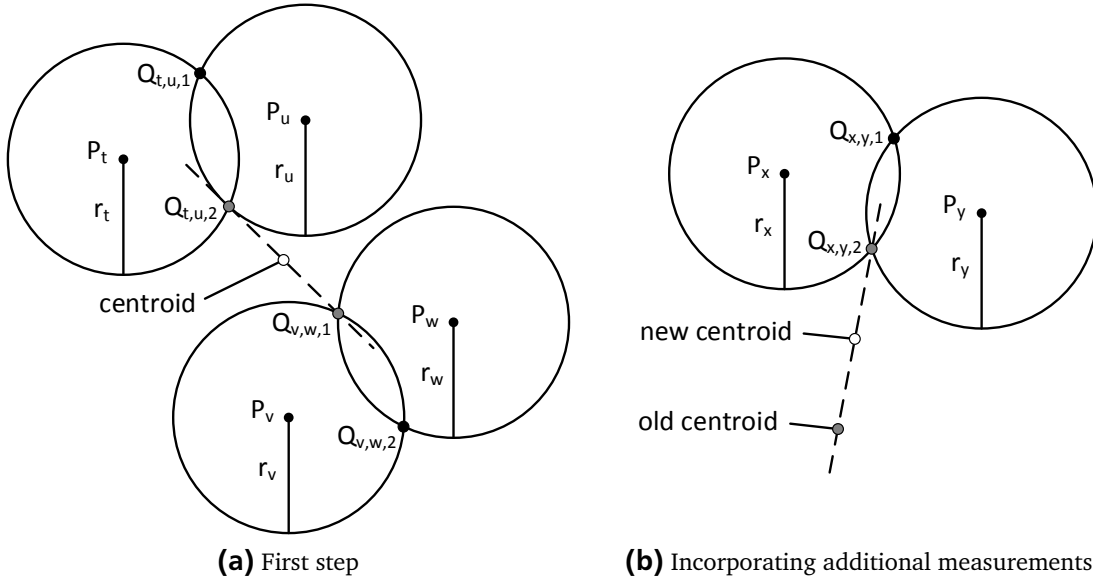


Figure 17: Multilateration

To incorporate all other points of the measurements we pick the next two arbitrary points P_x and P_y with intersecting circles and calculate the intersection points $Q_{x,y,1}$ and $Q_{x,y,2}$, as well. Using the point closest to the old centroid, we calculate the new centroid half way between both points (cf., Figure 17b). We iteratively proceed with all pairs of points with intersecting circles, thus, further correcting the estimation of the location for individual wireless routers.

Subjective Comparison

To demonstrate the difference between a data set obtained from WiGLE and a data set enhanced with our method, we briefly and subjectively compare both data collection methods. As described above, we use wardriving to collect data about wireless routers in the city center of our hometown Darmstadt and enhance the data set with multilateration. We collected data from both sources for the same area to compare them visually.

Figure 18 presents the results of this comparison. The picture of the WiGLE data set supports our assumption that most routers are placed on the streets rather than on the real locations (cf., Figure 18a). Our multilateration method seems to provide a more accurate estimation of the real router locations (cf., Figure 18b). In this figure, public routers are represented by red markers, routers with WEP encryption are depicted in blue, and WPA and WPA2 encrypted routers are represented by the black markers.

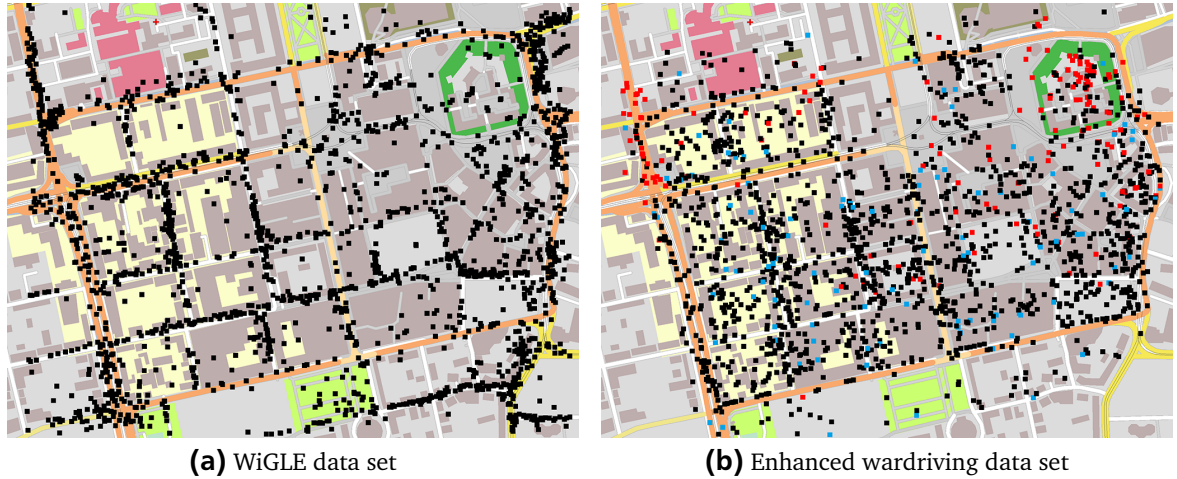


Figure 18: Comparison of the WiGLE data and our wardriving data in Darmstadt

4.2.3 Construction of CityMesh Networks

After collecting data about public or private wireless routers in cities, the second step is to virtually create CityMesh networks to analyze them in the third step. For this purpose, we use the unit disk graph approach [53]. In a unit disk graph, two nodes are connected only if they are no further apart than one distance unit. In our creation process, we simply use the distance unit r with different values. This creates differently connected CityMesh networks for our analysis. With higher r the resulting CityMesh network has more connections between routers. Figure 19 depicts an example unit disk graph. Circles around nodes have a diameter of r (radius of $r/2$), thus, nodes with touching circles are connected to one another.

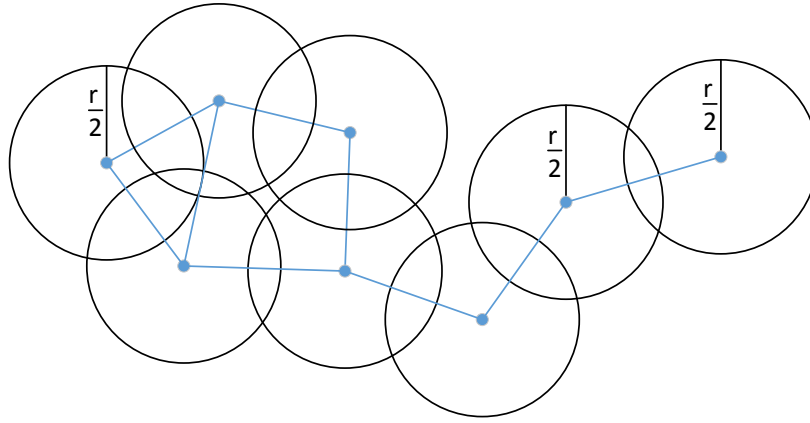


Figure 19: Example of a unit disk graph with distance unit r

4.2.4 Graph-theoretical Analysis of CityMesh Networks

To analyze the different graph-theoretical properties of the CityMesh networks, we use the Graph-Theoretic Network Analyzer (GTNA) by Schiller et al. [172]. It is a Java-based framework that allows for the graph-theoretic analysis of arbitrary network topologies. For this purpose, it provides a large set of standard graph metrics. It is possible to either import snapshots from network simulators or to generate popular network topologies. A large set of commonly used graph metrics and network topology generators are already provided by GTNA. The framework is also extendible through a well-defined plug-in interface. Furthermore, GTNA also has a plotting module which allows for a graphical visualization of the analyzed graph metrics. This helps us to analyze CityMesh networks for their connectedness, their resilience against individual node failures, and the possible routing performance. For this analysis, we use the following metrics:

Connectedness: We analyze the **average node degree** and the **degree distribution** for all networks to evaluate the connectedness of the respective graphs. The node degree is defined as the amount of links that are connected to one node. The average degree is then calculated as the average over all node degrees in one network. If a node has a node degree of 0, the node is called *isolated*. Networks with high average node degree are strongly connected. This is also one indicator for the resilience against individual node failures.

Resilience against node failures: We analyze the **maximum cluster size** and how the cluster behaves when removing critical nodes to evaluate the resilience of CityMesh networks. A cluster is a set of interconnected nodes and all nodes in a cluster have a degree greater than 0. A most critical node is defined as a node with the highest node degree. Such a node is also called a *hub*. By removing such nodes, many links of the network break and the average node degree decreases. If the maximum cluster size is not severely affected by the removal of multiple hub nodes, we assume the network to be resilient against individual node failures.

Expected routing performance: We analyze the **characteristic path length**, **shortest path length distribution**, and the **diameter** of the networks to evaluate the expected rout-

ing performance. The characteristic path length is defined as the average shortest path between all node pairs in the network. The shortest path length distribution is defined as the distribution over the shortest paths between all pairs of nodes in the network. The diameter is defined as the longest path over all shortest paths between all node pairs in the network. High characteristic path length values indicate a bad routing performance because routing paths tend to become long. On long paths, increased packet loss probability as well as high packet delay are to be expected when transmitting packets. However, it also means that the network is capable of covering a large area and connections can be established over long geographical distances.

4.3 Feasibility Evaluation of CityMesh Networks

We introduced our concept of the CityMesh and the emergency switch for wireless routers. Once activated, the switch would transition the routers into emergency mode, thus, creating the CityMesh network by interconnecting all routers in emergency mode. We also described our methodology to evaluate the feasibility of potential CityMesh networks by means of collecting and analyzing data of public and private wireless routers. In the present section, we discuss the results from our evaluation.

Throughout our evaluation of CityMesh networks, we pursue two main goals. First, we demonstrate the difference between random network topologies and potential CityMesh networks using public routers. In most simulation studies of wireless mesh networks, nodes are randomly placed on a square or rectangular field [146]. We believe a random placement of nodes does not reflect the reality. Most cities have city centers or mall areas where most shops, bars, and restaurants are resided leading to a higher density of public routers in such areas. We examine graph-theoretical properties of both the randomly generated mesh networks and the CityMesh networks derived from routers in major cities. Thereby, we show a CityMesh network to differ significantly from randomly generated networks in these properties. Furthermore, resilience against individual node failures differs, as well.

Second, we analyze the impact of private routers on the resilience of potential CityMesh networks. Due to the high number of private routers in cities, we assume that even low communication ranges result in highly resilient networks. We, again, analyze the resulting networks for their graph-theoretical properties. We demonstrate the drastic impact in the resilience of potential CityMesh networks if private routers are incorporated into the network.

4.3.1 Random Network Topologies vs. CityMesh Networks

We have conducted an analysis of CityMesh topologies in five different cities that are covered by the NodeDB project (cf., Section 4.2.1). We selected the following cities for our evaluation: Chicago, Melbourne, Mountain View, New York, and San Francisco. Mountain View is especially interesting as Google Inc. deployed a city-wide WiFi network to provide free Internet access (cf., Section 3.1.3). As introduced earlier, routers in that database were published voluntarily. Hence, we assume all routers in that database to be public routers, e.g., from owners letting their customers know where to get access to the Internet.

To create the CityMesh networks with the unit disk graph approach (cf., Section 4.2.3), we use a communication range of $r = 300m$ for all cities¹⁷. Furthermore, we generate random network graphs for each city using the same properties (number of nodes, dimensions of the respective area, communication distance r). Due to the high density of bars, stores, restaurants, etc. in city centers, we expect important graph metrics to be different from randomly generated unit disk graphs. The city center contains more nodes and also more links between the nodes than outside regions. By looking at Manhattan, we can see that places like parks (i.e., the Central Park) do not have any routers at all whereas the city center is strongly connected (cf., Figure 20e).

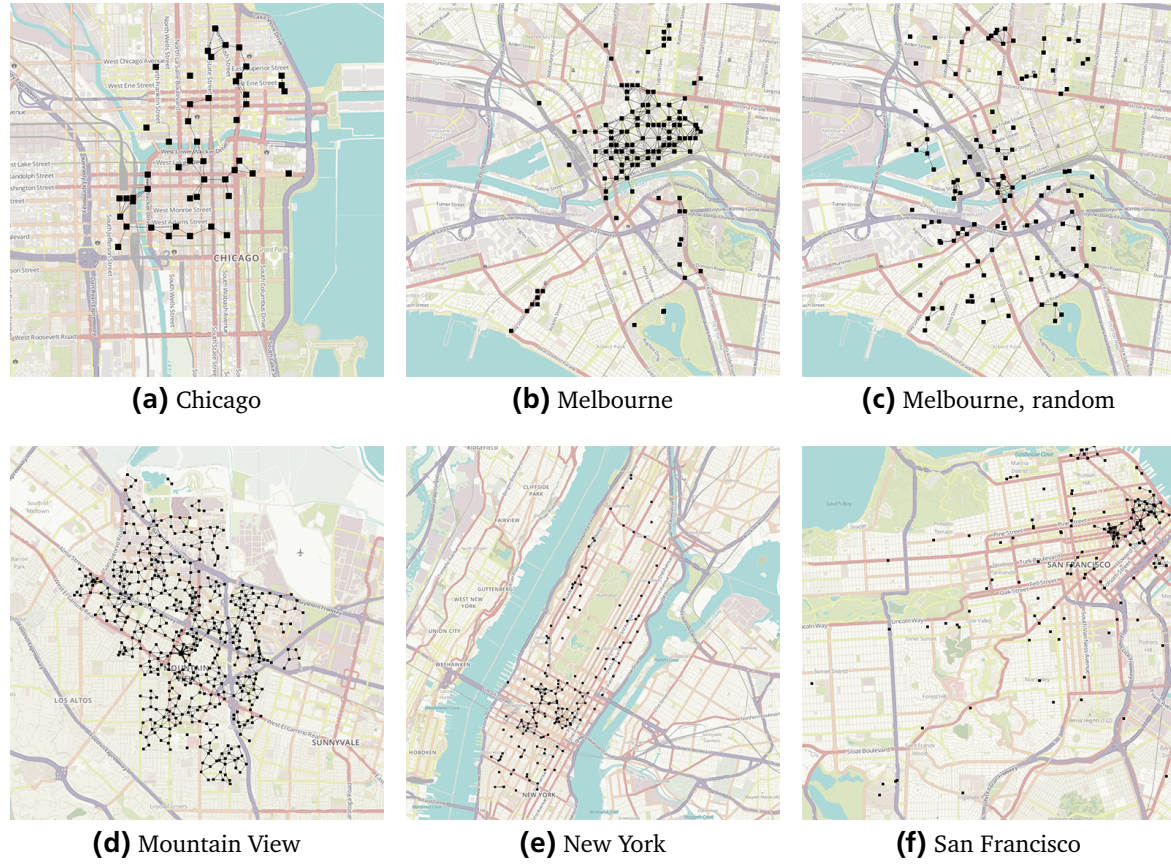


Figure 20: CityMesh networks of public wireless routers in big cities

Figure 20 shows the CityMesh networks in the examined cities. It also demonstrates the difference between the CityMesh network in Melbourne (Figure 20b) and a randomly generated network exhibiting the same basic properties (Figure 20c). The city center contains most of the routers, the nodes are strongly connected, and the link count is much higher than anywhere in the random network. Also, there are huge spaces in the CityMesh network without routers. Both facts are due to the geographical and architectural characteristics of the city. In the random network, the nodes are distributed equally over the entire area.

¹⁷ Hardware vendors often claim a communication range of 300 meters for their devices. Similar values are assumed in simulation-based experiments. Zhu et al., for instance, assume a value of 250 meters [207].

Table 7 compiles a list of all important properties of the five evaluated cities. The table lists the size of the considered area, the number of collected routers in that area, and the router density. All cities exhibit different properties. We, therefore, compare the CityMesh network against randomly generated networks with the same basic properties provided by the table. This helps us finding a baseline for each city and, at the same time, enables us to compare the cities with each other. For statistical significance, we generate 20 networks, calculate average values for every metric, and compare the results against the respective CityMesh network.

	Height (m)	Width (m)	Area (km ²)	#routers	$\frac{\#routers}{km^2}$
Chicago	2577	1998	5.15	49	9.51
Melbourne	5105	3126	15.96	115	7.20
Mountain View	5572	6613	36.85	488	13.24
New York	2964	10898	32.30	162	5.02
San Francisco	5047	5246	26.48	136	5.14

Table 7: Properties of all analyzed cities

Connectedness

Table 8 provides the average node degree values for all generated CityMesh networks and the respective random networks. Melbourne has the highest average node degree due to the very high router density in the city center (cf., Figure 20b). In their corresponding random graphs, San Francisco and New York have very similar average node degree values due to the similar router density (cf., Table 7). But their actual average node degree values differ since the cities have different shapes (cf., Figure 20). In fact, we can see that cities have unique average node degrees for that matter. Only Chicago has a rather similar average node degree compared to the corresponding random network. The average node degree of the randomly generated graphs, however, are proportional to the router density ($D_{avg} \approx \frac{1}{2} \cdot \frac{\#routers}{km^2}$).

D_{avg}	Chicago	Melbourne	Mountain View	New York	San Francisco
City	5.06	13.32	9.53	5.13	7.67
Random	4.38	3.78	7.16	2.67	2.65

Table 8: Average node degree values for evaluated cities

Taking a more detailed look on the CityMesh networks of Melbourne and Mountain View, we see that in Melbourne there are many interconnected routers in the city center. In Mountain View, on the other hand, Google Inc. deployed routers equally in the city to achieve good network coverage throughout the city. Figure 21 depicts a comparison of the degree distributions of Melbourne and Mountain View. Since the degree distribution of Melbourne is similar to the distribution of the other cities, Melbourne can represent them in this analysis. Both plots in Figure 21 depict the degree distributions of the CityMesh network (red)

and of the random network (blue). In both cities, we observe that the degree distribution of random networks hits a peak value at around 1 for Melbourne and around 3 for Mountain View. After that peak, the probability of nodes with high degree decreases.

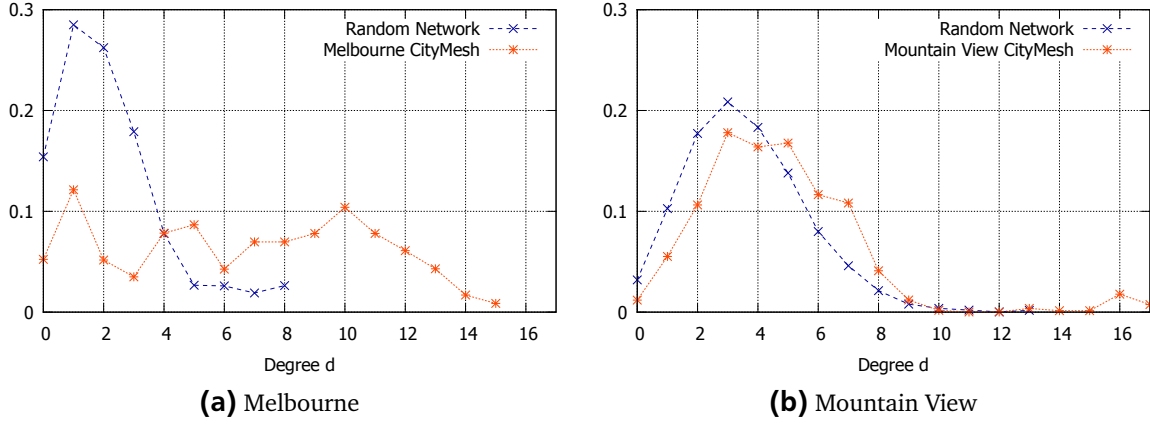


Figure 21: Degree distribution of Melbourne and Mountain View

In the mesh network of Melbourne, we cannot identify a real peak value and the probability of different node degrees seems to be distributed arbitrarily over all node degrees. The probability for nodes having a node degree of more than 10, however, decreases again. This results in a high average node degree in Melbourne. In other cities, the node degree distribution is similar but shifted towards lower values.

In Mountain View, the node degree distribution does not differ significantly from the corresponding random network. Also, the average values are closer to each other than in Melbourne. We believe this results from the high router density in Mountain View and the geographical distribution of routers in the city. There are nearly twice as much routers per km^2 in Mountain View than there are in Melbourne. Google tries to reach high network coverage for the entire city and, therefore, distributes routers equally in the city. We will see later, however, that the nodes are not randomly deployed.

Resilience Against Node Failures

Communication infrastructure in harsh environments must be resilient to node failures. To evaluate the resilience of the biggest cluster, we stepwise remove hub nodes as stated before. Again, we concentrate on Melbourne and Mountain View since they exhibit the most interesting characteristics. While the network in Mountain View is planned by Google Inc., the networks in Melbourne and in the other cities emerged more occasionally. The biggest cluster of the Mountain View network consists of nearly 460 nodes (the overall network consists of 488 nodes). That translates to almost 95% of all nodes leaving only very few nodes isolated. In Melbourne, on the other hand, only 83 out of 115 nodes (72%) belong to the biggest cluster. When it comes to network communication, most of the traffic will be passed through such hub nodes. Incrementally removing them shows us how fast the network will fall apart if the routers would run on battery. The results of removing these hubs are depicted in Figure 22.

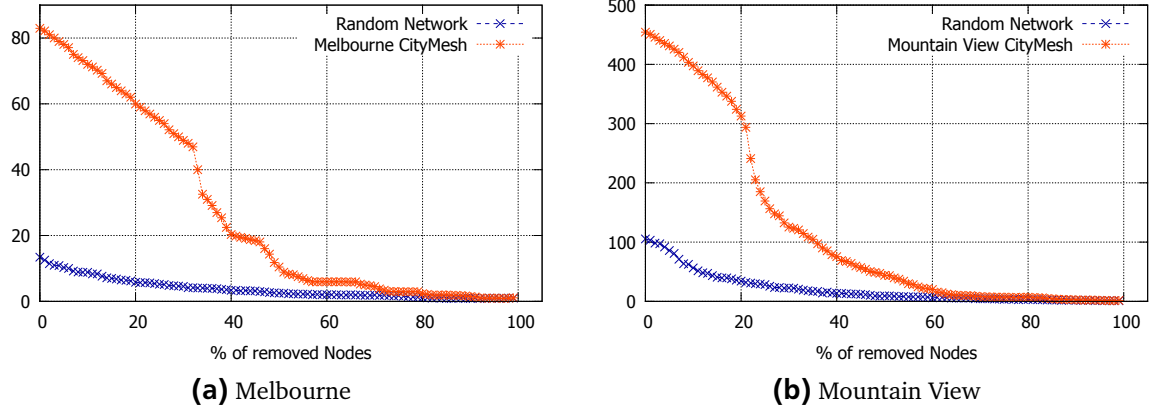


Figure 22: Maximum cluster sizes in Melbourne and Mountain View when removing hubs

In both networks, removing hub nodes decreases the size of the biggest cluster since most of these nodes are part of that cluster. By removing 20% of these hubs, the network of Mountain View is still connected. That is a very high value and translates to 72 nodes that need to fail. When we remove more nodes, the size of the biggest cluster of the Mountain View network drops drastically. This means that at this point the biggest cluster is partitioned resulting in two or more clusters with smaller size. In comparison, the network in Melbourne is still connected with over 30% of the hubs removed. After that, we again observe a partitioning of the biggest cluster. If we compare both networks to their random baseline, we can see that the biggest cluster is much smaller in size. This also means that the network most probably consists of many small clusters and even isolated nodes.

Expected Routing Performance

The characteristic shortest path length (CPL) is a good indicator of the average hop count for a message to route through the network. Although high values seem to describe bad routing performance, it is important to keep in mind that high values also mean participants can communicate over a long geographical distance using the CityMesh network. The values in Table 9 have to be evaluated under both aspects. In general, random networks exhibit smaller characteristic shortest path length values. This is because the resulting networks consist of many small clusters all with small diameter. In such a network, participants have to be in short distance to still be able to communicate with each other.

CPL	Chicago	Melbourne	Mountain View	New York	San Francisco
City	3.14	3.71	14.29	5.84	3.74
Random	2.62	2.26	8.43	1.97	1.73

Table 9: Characteristic shortest path length (CPL) of CityMesh networks

Again, Mountain View exhibits the longest characteristic path length. While this might lead to higher message delay due to the large number of hops, we have to consider that

the biggest cluster covers almost the entire city. Hence, it is possible to communicate over a very long geographical distance. This fact is also supported by Table 10 that provides values for the diameter of the network. In Mountain View, routes can become up to 39 hops in length which translates to roughly 9 km distance between the two nodes. The corresponding random network exhibits similar high values but does not reach the performance of the real Mountain View network. Although one would expect that the equal distribution of routers in the city would perform similar to the corresponding random network, these values show that the Mountain View network is not randomly deployed but rather planned.

<i>Diam</i>	Chicago	Melbourne	Mountain View	New York	San Francisco
City	9.0	9.0	39.0	17.0	10.0
Random	6.7	6.9	27.85	6.3	5.15

Table 10: Diameter of CityMesh networks

Regarding the other cities, we observe that especially New York and San Francisco have a very high skew in their results compared to their corresponding random network. The reasons for this are the geographical properties of the respective area. In the city center with many bars, restaurants, and shops, there is a high concentration of routers. Whereas in other spots, such as the Central Park, there are no routers at all. It is also important to note that Chicago exhibits results similar to the corresponding random network. We have already mentioned Chicago to have the most random-like distribution. However, the results are still significantly different.

4.3.2 Impact of Private Routers

Above, we analyzed CityMesh networks using public routers in large cities and on a large scale. In the present section, we discuss our evaluation results of CityMesh networks in a smaller urban area and on a smaller scale. For this purpose, we use our data set of wireless routers in our hometown Darmstadt that we previously collected with the wardriving method. The data set contains so many wireless routers so that a much smaller communication range can be assumed to create connected mesh networks. Below, we use communication ranges r with $r \in \{10, 20, 30, 40, 50, 60\}$ meters, therefore, creating six different graphs. We distinguish between non-encrypted (public routers) and encrypted networks (private routers) to investigate the impact of private routers on the CityMesh network. Figure 23 shows how the different CityMesh networks would look like when constructed with $r = 30m$ or $r = 60m$ using only public routers or all routers. In the following evaluation, we show that in the city center of Darmstadt enough wireless routers exist to create a resilient emergency mode mesh network even if we assume a rather low communication range of $r = 30m$.

Connectedness

We investigate CityMesh networks constructed with all wireless routers in Darmstadt. As expected, both the average node degree and the maximum node degree increase continu-

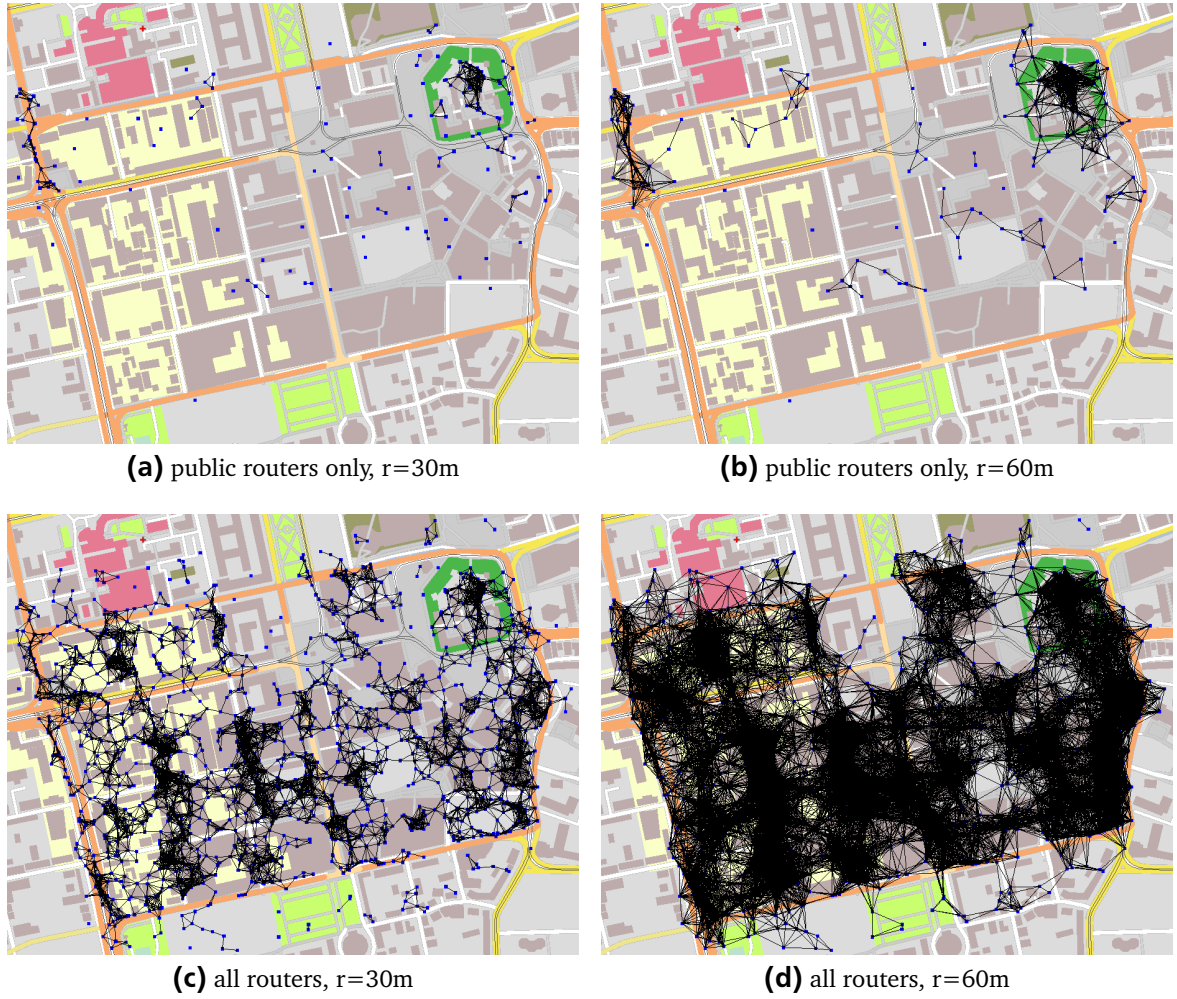


Figure 23: Constructed mesh networks in Darmstadt

ously with increasing communication range r (cf., Table 11). Also, the number of isolated nodes decreases with an increasing communication range (cf., Figure 24). With a range of $r = 10m$, about one third of all nodes are isolated. With a range of $r = 20m$, less than 7% of all nodes are isolated. However, with a range of $r = 30m$ only 2% of all nodes are isolated and with $r = 60m$ no node is isolated at all. With an increasing communication range r , the distribution of the node degree gets further stretched out. This means that the resulting network becomes even more interconnected because one router can reach more neighboring routers directly.

Table 11 presents the metrics for the different networks. In the network constructed with a range of $r = 10m$, the average node degree is 1.324. We can say that on average every node has roughly one neighbor. In networks with a range of $r = 30m$, one node has about 10 neighbors on average and with a range of $r = 60m$ a node has almost 36 neighbors on average. We believe a communication range of $r = 30m$ to be realistic in an urban area due to buildings and other obstacles reflecting and interfering with wireless signals. Such networks have only 2% isolated nodes and about 10 neighbors per node on average and a

Range r	10m	20m	30m	40m	50m	60m
Edges	1010	3619	7874	13209	19700	27165
Avg. degree	1.324	4.74	10.313	17.301	25.802	35.58
Max. degree	8	16	28	42	54	68
CPL	1.204	1.864	2.43	3.427	4.426	4.698
Diameter	4	8	10	17	20	19

Table 11: Network properties for different communication ranges r

maximum of 28 neighbors. This makes such CityMesh networks very suitable for disaster relief communication.

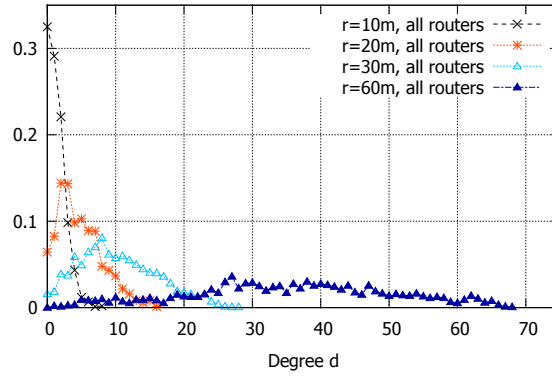


Figure 24: Degree distribution for CityMesh networks in Darmstadt

Resilience Against Node Failures

To analyze the network resilience against failures of hub nodes, we investigate the amount of clusters in the networks. Figure 25 depicts how the amount of clusters behaves when removing the most important nodes, the so called hubs. Assuming a range of $r = 10m$, the network consists of many clusters in general. In the initial state without removing any nodes, the network is already partitioned into almost 800 clusters. But with $r = 20m$ the number of clusters already decreases drastically to less than 200 clusters. A range of $r = 30m$ decreases the number of clusters even further resulting in about 50 clusters in the initial state. When we remove hub nodes, we see an increase in the number of clusters almost immediately regarding networks generated with a range of $r = 10m$ and $r = 20m$. But with a range of $r = 30m$, the network is highly interconnected so that even removing 16% of hub nodes does not affect the number of clusters. Assuming a higher range amplifies this effect even more.

Removing these hubs also has an impact on the biggest cluster in the network. Figure 26 shows maximum cluster sizes when successively removing hub nodes from the network. After removing 10% of hub nodes from the network with $r = 20m$, we see the first drop in the maximum cluster size meaning the partitioning of the biggest cluster. With a communication range of $r = 30m$, we can remove up to 22% hub nodes until partitioning of the biggest

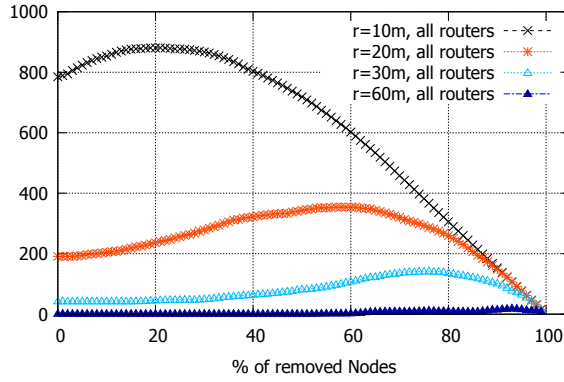


Figure 25: Number of clusters when removing hubs in Darmstadt CityMesh networks

cluster happens. With a range of $r = 60m$, this partitioning happens after removing 62% of hubs.

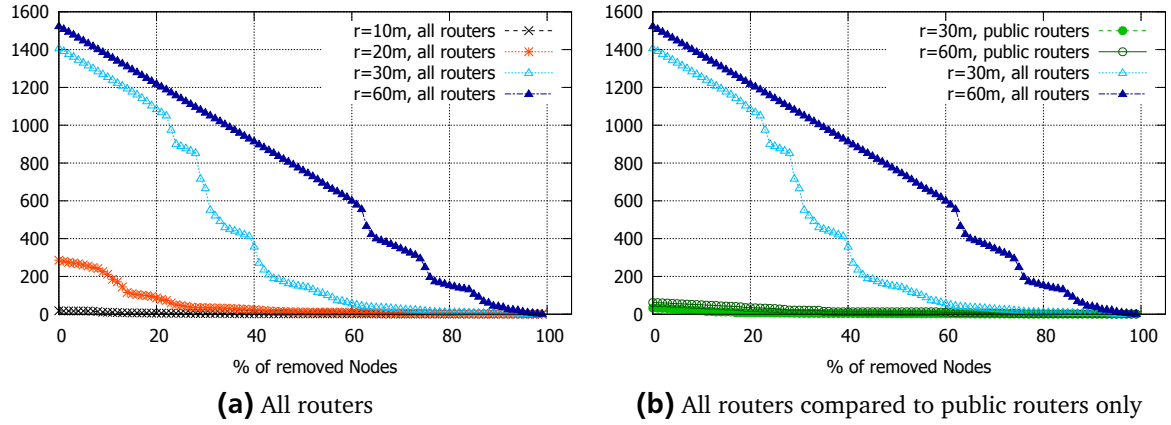


Figure 26: Maximum cluster sizes when removing hubs in Darmstadt CityMesh networks

Expected Routing Performance

The shortest path length distribution shows how the shortest paths are distributed over all nodes in the network. It can be seen as an indicator what routing performance can be expected from the resulting networks. Figure 27 presents our results of the shortest path length analysis. It seems as if a network constructed with a range of $r = 10m$ has the best routing performance because more than 80% of all node pairs have a shortest path length of one hop. As presented in Table 11, the characteristic path length is 1.204 hops and the diameter is only 4 hops. But, when recalling the number of clusters (cf., Figure 25), we see that such a network has almost 800 isolated nodes that cannot be reached at all. With a communication range of $r = 30m$, 33% of node pairs have a shortest path of one hop but only 50 clusters exist in total and the biggest cluster consists of over 71% of all nodes. The characteristic path length in this network is 2.43 hops and the diameter is 10 hops. With increasing communication range the diameter and the characteristic path length also

increase further. This can be seen as a drop in routing performance since the number of reachable nodes does not increase as much as the average path lengths. A range of $r = 30m$ is also a good tradeoff between routing performance and network resilience as described before.

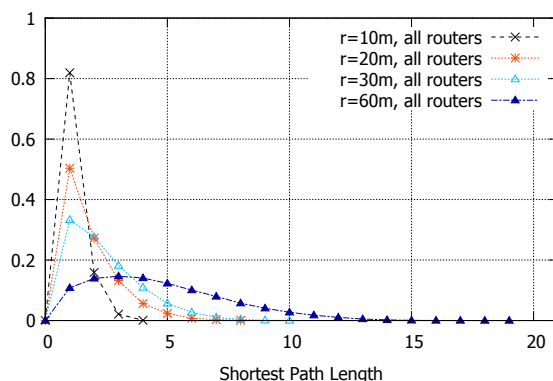


Figure 27: Shortest path length distribution in Darmstadt CityMesh networks

Comparison with Public Networks

As we compare the networks constructed with all routers to the networks generated from public routers only, we can see a big discrepancy between them. Our results indicate such public networks to have about 30% of isolated nodes when generated with a range of $r = 30m$. This share is 15 times greater than compared to the network generated from all routers. Even with a communication range of $r = 60m$ the resulting network still has about 5% isolated nodes.

Furthermore, Figure 26b shows how small the public networks are compared to the networks constructed with all routers. In public networks with $r = 30m$, less than 20% of all nodes are connected to the biggest cluster. Whereas more than 71% of all nodes are connected to the biggest cluster when private routers are also included into the network creation process.

In public networks with $r = 30m$ and $r = 60m$, about 60% of all node pairs are connected with only one hop. The longest path has 5 hops in both cases due to the low node density. And because less than 20% of all nodes can be reached, the networks with only public routers also have bad routing performance in terms of reachability. These results indicate that private routers drastically improve not only network performance but also resilience in an emergency mode mesh network.

4.3.3 Conclusion

Overall, our evaluation shows that CityMesh networks have very different graph-theoretical properties compared to corresponding random networks. The average node degrees as well as the node degree distributions are unique for each city. Geographical and architectural characteristics are responsible for the unique router distribution within individual cities. We have seen that public routers tend to occur in certain areas with higher density (i.e., city

centers). This means that random network topologies are not suitable for the meaningful simulation-based investigations of wireless mesh networks. Random topologies can neither represent realistic topologies as found in cities nor pre-installed and working mesh networks (as found in Mountain View, for instance).

In Mountain View, Google Inc. installed a wireless mesh network in the entire city to provide free Internet access. Wireless mesh routers are mounted on utility poles, e.g., light poles. The network consists of more nodes than the other public networks. It further exhibits high resilience against node failures and good routing performance in terms of reachability and network coverage. Nevertheless, all public networks require a rather high communication range ($r = 300m$). This can only be achieved under very good conditions and requires line of sight between individual routers. In real cities, however, this cannot be assumed because routers are usually located inside buildings.

We, therefore, evaluated the impact of private routers on the resilience and network coverage of CityMesh networks. In the city center of Darmstadt, we found plenty of private routers so that a communication range of only $r = 30m$ would be sufficient to create highly resilient CityMesh networks covering the entire area. Finally, we conclude that private wireless routers are inevitably required to create a wireless ad-hoc communication infrastructure for disaster relief in urban areas. Only with all available routers, a resilient network covering the entire city can be created.

4.4 Discussion of Realization Aspects

Our evaluation of several CityMesh networks indicates a vast improvement in network resilience if privately owned wireless routers would participate in emergency networks. But, including these routers has also major implications that we want to discuss in this section. We argue that an emergency switch would be possible today by creating a separate guest network. This could be leveraged in case of a disaster event. A simple firmware update could integrate the emergency switch into routers today. Assuming this would be realized, several questions arise:

- How could power be supplied to wireless routers, if the electricity is influenced by the disaster? How long would such a power supply last?
- Who would actually be able to switch routers into the proposed emergency mode? What would an implementation of the emergency switch look like?
- How could both networks be secured and isolated from each other?
- What is the legal basis of such an emergency switch and would it be accepted by citizens?

In the following sections, we will give first answers to these questions.

4.4.1 Energy Supply

The most severe issue in disaster situations is the energy supply. How could power be supplied to wireless routers if the electricity is influenced by the disaster? How long would

such a power supply last? Besides Internet and telephone connections also electricity is directly influenced by incidents. Earthquakes, floods, and other disasters may destroy the electrical infrastructures of urban areas. Often, entire blocks of buildings are left without electricity, although houses and apartments are not directly damaged by the disaster. A cut cable or a broken transformer may lead to electricity outage in an entire region. A study conducted by Jennex [97] after the Great San Diego Blackout of 2011 targeted at students of the San Diego State University and their friends and relatives revealed that around 64.6% lost their Internet connectivity completely and 26.9% reported a degraded Internet service. Only 8.6% of those polled in the survey had no loss of Internet connectivity. Even if the proposed emergency switch would exist in routers, only less than 10% might function normally. How could the energy supply be ensured after a disaster incident so that coverage of the emergency mode network would not be decreased?

We argue that manufacturers should equip new routers with batteries to ensure several hours of emergency communication once the electricity supply breaks down. Furthermore, additional energy supplies could be used such as solar panels [189]. To further improve the lifespan of the emergency network in such situations, the citizens' home network should be shut down as well as all other non-emergency-related activities of the routers. Basically, this means that only the emergency network would be supplied with energy from batteries during an outage of electricity. Such precautions require a regulation by the government to ensure new wireless routers are equipped with additional energy supplies. Finally, additional infrastructure can be rolled out to support the CityMesh network where parts of the network failed due to electricity outage (cf., Section 3.1.2).

4.4.2 Integration and Activation

The following questions need to be answered concerning the integration and activation of the proposed emergency switch. Who would actually be able to switch routers into the proposed emergency mode? What would an implementation of the emergency switch look like? Although our proposed emergency switch could be integrated into private and public routers on a voluntary basis, we believe the switch should be implemented by default. However, there could be the possibility to opt-out on the emergency switch. Users who do not want their routers to be switched into emergency mode could then configure it to disable the emergency mode switching. We believe, however, that most users would not disable the emergency mode switching. Many users would simply not care about the switch or would even support it. Furthermore, most users would not know about it or would not have the technical experience to change the required settings.

The activation of the emergency mode should be done by first responders or the government before or at the beginning of the disaster recovery mission. The easiest way to activate the emergency mode network would be to directly activate it through the Internet. However, Internet connection might not be available due to destroyed or overloaded communication backbone. Therefore, another activation method should exist. First responders could activate the emergency mode network with their communication devices by sending a special beacon signal to surrounding wireless routers. Routers receiving this signal would then transition into the emergency mode creating the proposed network. However, both methods must be secured from abuse at all cost.

4.4.3 Security and Isolation of Networks

Security is a rather big issue of our proposed method. How could both networks be secured and isolated from each other? Not only the activation mechanism but also the emergency mode network itself must be secured from infringements from the outside as well as from the inside. First off, authentication mechanisms are required so that only authorized emergency personnel can access the emergency switch and the emergency network. Furthermore, encryption is required so that exchanged messages sent through the emergency network cannot be read or changed by unauthorized individuals. Finally, the privacy and the rights of citizens must be protected. Therefore, the emergency network should be isolated from the home networks of citizens and the home network communication must be encrypted with strong encryption techniques, as well. Most of these problems can be solved already today with state of the art authentication and encryption schemes.

To further increase the difficulty for attackers to access the emergency switch and the emergency network, we argue that a separate router module dedicated only to emergency mode networking should be installed in the router. Furthermore, both networks should operate on different frequencies. This could help separating both networks also physically and would increase security of both networks. However, this would also require the manufacturer to directly build in the module for emergency networks. Thus, emergency mode networking could not be realized with installed hardware of today's households. Also, by physically separating both networks to increase the security, first responders could no longer utilize the Internet connection of private households.

During our measurements, we observed around 11% of all routers in the city to be missing encryption and around 7% of all routers to be still encrypted using the outdated WEP encryption technology. Several works showed that WEP encryption is easily compromised [187, 188]. Furthermore, we believe that many pass phrases for wireless encryption generated by users could easily be broken. Studies showed that users tend to use simple passwords for email accounts [65], for instance. We believe the same holds true for pass phrases of wireless networks and, therefore, an emergency switch would not introduce more vulnerability to a user's home network than the user already introduces by himself. Instead, the discussion about the emergency switch could be used to educate people about possible security risks.

4.4.4 Legal Issues and Societal Acceptance

Finally, there are legal issues and concerns about the societal acceptance of our proposed approach. What is the legal basis of such an emergency switch and would it be accepted by citizens? At first glance, enforcing the emergency switch in private routers by law would not be easily established in Germany. This has mainly two reasons. First, the supremacy of citizens over their own wireless routers would be compromised if the government would be able to remotely activate the emergency switch. Second, the remote activation of the emergency mode network from the outside without the knowledge of the owner is a severe intrusion into the owner's privacy.

We believe this is a problem of the perceived risk. After the cold war ended, Germany repealed a law that enforced the construction of bomb shelters because the perceived risk of

another world war breaking out was very low. Switzerland, however, still has an extensive network of bomb shelters in public as well as private buildings. Furthermore, there is already a proposal to connect cars equipped with eCall to Germany's emergency broadcast system to alert citizens in case of a crisis by remotely activating horns of parking cars [89]. We believe this approach and our proposal share a similar legal basis in terms of remote activation. If society and politics would perceive the risk of disasters and the risk of communication outage during rescue missions as high enough, a law enforcing the emergency switch would be quickly established. Raising awareness is, therefore, the first step towards an emergency switch for wireless routers.

Finally, implementing the emergency switch on a voluntary basis is another approach we want to investigate briefly. Large scale social media exercises like X24 show how supportive people all over the world are when it comes to crisis response [88]. The exercise consisted of two parts. X24-Europe had over 49,000 participants from at least 92 countries. We, therefore, believe that many people are willing to share their routers and would participate in an emergency network even on a voluntary basis as long as their privacy remains unaffected. Nevertheless, this approach requires further investigation using surveys to get a clear picture about the motivation and the attitude of citizens. We see this approach only as an alternative to a regulation by the government.

4.5 Summary

In this chapter, we introduced our general concept of the *CityMesh*, a privatized pre-deployed communication infrastructure for decentralized first responder communication. This infrastructure is created by interconnecting wireless routers found in private households and in public institutions, thus, forming a wireless mesh network. In a final implementation, we imagine wireless routers to have an emergency switch integrated that could be triggered by an authorized institution in case of a disaster. This would then transition wireless routers into emergency mode and create the communication infrastructure.

We presented our methodology to evaluate the feasibility of such CityMesh networks. Our methodology consists of three steps: data collection, network construction, and network analysis. The presented data collection methods include online database access as well as measurements of existing wireless routers using wardriving. We further enhanced data collected with wardriving by using multilateration on gathered data points. This estimates the real locations of wireless routers more accurately than the footprinting method used in online databases. We constructed networks with the unit disk graph approach and analyzed them for their graph-theoretical properties using the GTNA tool.

In our evaluation of the feasibility of CityMesh networks, we showed that networks in five selected cities differ significantly from randomly generated networks in several graph-theoretical metrics. The geographical and architectural characteristics of a city (e.g., population density, locations of shops, locations of parks, etc.) play an important role in the distribution of wireless routers throughout the respective city. One of our main findings is that the CityMesh networks can cope with the removal of 20% of the most critical nodes (hubs) without partitioning. Furthermore, we found the density of (private and public) wireless routers in the city center of Darmstadt to be very high. If using all routers to create CityMesh networks, a relatively low communication range of only 30 meters is sufficient to create resilient networks covering the entire city center. Reducing communication range

safes energy consumption of routers. If routers are to be equipped with batteries, the reduced communication range can result in long-lasting CityMesh networks even during an outage of electricity.

Finally, we discussed several aspects related to the realization of an emergency switch for wireless routers. We pointed out different questions that have to be answered before such an emergency switch can be realized. Although our proposed emergency switch for wireless routers might be perceived as a privacy infringement, we believe this concept to be necessary to successfully assist rescue workers in their missions. We, therefore, believe a regulation by the government is required to effectively realize our CityMesh concept.

In the next chapter, we present our peer-to-peer service overlay to provide versatile services in harsh environments. Due to the high resilience against node failures, we assume CityMesh networks to be used in the underlay network to some extent.



5 Peer-to-Peer Service Provision

Service provision in harsh environments requires a resilient communication infrastructure (cf., Chapter 4) and a peer-to-peer-based system, the service overlay (cf., Section 2.3). We identified the service placement to be the most crucial component of our proposed service overlay concept to deliver high quality services under dynamic conditions. Service placement relocates service instances, thus, introducing service mobility through service migration. Most service placement approaches aim at the reduction of overall network traffic, not considering the quality of provided services (cf., Section 3.2.3). Furthermore, state of the art service discovery approaches are not explicitly designed to support service mobility (cf., Section 3.2.5). We, therefore, focus on these two components. The contributions in the present chapter are threefold:

1. We propose four service placement approaches using hop count and latency as quality metrics. We investigate all four placement approaches in a disaster recovery setting using static placement and a state of the art service placement as baseline (cf., Section 3.2.4). We demonstrate that our approach achieves consistent service quality for all connected service consumers.
2. We evaluate the performance of three selected state of the art service discovery approaches under service mobility introduced by service placement. We use global knowledge and two naive flooding-based approaches as baseline. We demonstrate that the selected state of the art approaches are not capable of handling the dynamics introduced by service mobility and that a simple request flooding approach can outperform them.
3. We present four concepts for smooth service migration. Two concepts target the elimination of interruptions during service migrations and two concepts target the reduction of generated data traffic. We integrate these four concepts in our real-world peer-to-peer service overlay prototype.

Parts of the contributions in the present chapter were published in the *Praxis der Informationsverarbeitung und Kommunikation* journal [148] and in our chapter “Service Overlays” [149] in “Benchmarking Peer-to-Peer Systems” [60].

The remainder of this chapter is organized as follows: In Section 5.1, we present our peer-to-peer service overlay concept in more detail. We focus on the design and implementation of the service placement and the service discovery components as well as on additional components required for our simulation-based investigations. In Section 5.2, we present our proposed service placement approaches based on the hop count and on the latency quality metrics. Furthermore, we investigate all four placement approaches in a disaster recovery setting. In Section 5.3, we present our evaluation of different service discovery approaches. We especially focus on the impact of service placement on the service discovery performance. Finally, we present our concepts for smooth service migration in Section 5.4 and summarize the chapter in Section 5.5.

5.1 System Model

In the present section, we describe the system model of our peer-to-peer service overlay in more detail. In the following sections, we first present an overview of our system model. This overview includes the two roles of peers: service provider and service consumer (Section 5.1.1). Second, we present the interplay between the two most important components (cf., Chapter 2 and Section 3.2): service discovery (Section 5.1.2) and service placement (Section 5.1.3). Finally, we describe additional components relevant to our simulation-based investigation of service placement and service discovery (Section 5.1.4).

5.1.1 Overview

Figure 28 depicts two peers with the extended system model of our proposed peer-to-peer service overlay. The first thing to notice is the additional container component, the service overlay core. This component contains all core components of the service overlay described in Section 2.3. Furthermore, this component provides an interface to access the service overlay functionality via the methods defined in Section 2.4: put, invoke, register, and access. The interface can be accessed by a user to consume a service in a real world implementation. In a simulation environment, the interface can be accessed by simulation components to trigger the consumption of services and other functionality for investigation purposes.

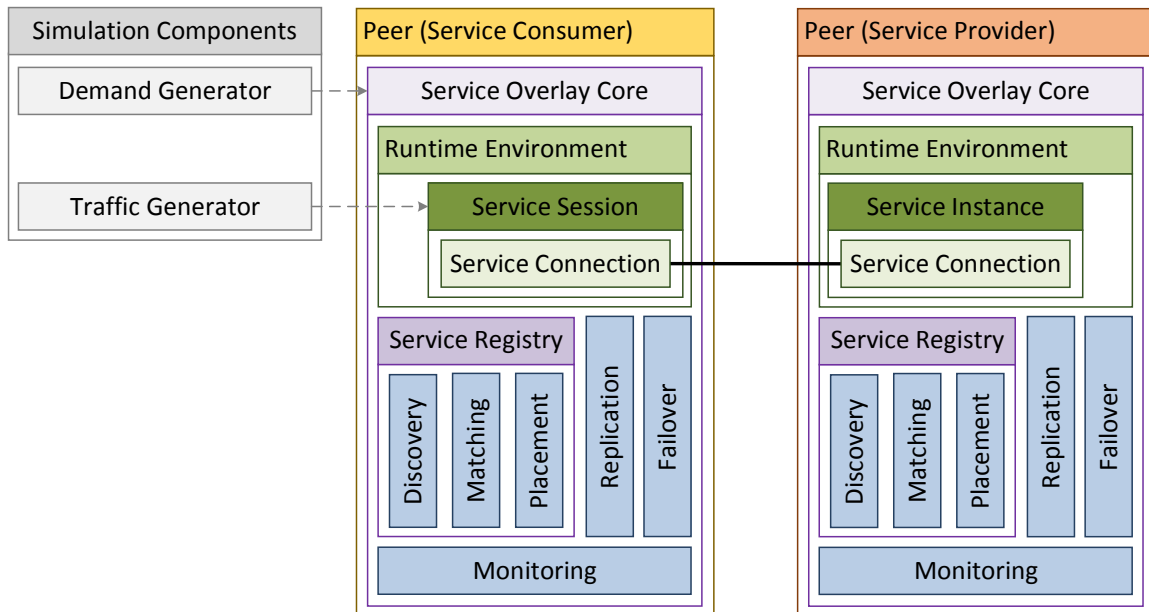


Figure 28: Extended system model of the service overlay, including simulation components

Furthermore, the figure illustrates how the service consumer interacts with the service instance hosted on the service provider. As discussed in Section 2.2, a service is a software component (i.e., application) that consists of a service instance and of a client interface stub. The service consumer can access the functionality provided by the (remote) service instance through the local client interface stub (called service session). The service session and the

service instance both use a service connection component to communicate with each other via the peer-to-peer network. For local service consumption, the service session and the service instance can be also executed on the same peer. Furthermore, the service instance can be migrated between peers as discussed in Section 2.3.

Finally, simulation components are necessary for our simulation-based investigation of several aspects of the peer-to-peer service overlay. In particular, two load generators are required imposing service-related load on the service overlay. First, the demand generator selects peers to act as service consumers for a specified amount of time. This component models the intention of participants to consume various services. Second, the traffic generator produces data traffic caused by a service consumer while consuming a service. This component models the behavior of participants when consuming a service. In the following sections, we discuss the interplay between the core components and describe the load generators relevant for our simulation-based investigation in more detail.

5.1.2 Service Discovery

Service discovery enables service consumers to discover their needed services. We present the classes relevant for service discovery in Figure 29. For better illustration, colors in the UML class diagram represent the class types: interfaces are blue, (abstract) classes are green, and enumerations are brown. As described in the previous section, a potential service consumer contacts the `SERVICEDiscovery` component through the interface provided by the `SERVICEOverlayCore`. Once the service discovery locates the service of interest, it delivers it back to the user through the `SERVICEOverlayCore`. The service consumer can then instantiate the local `SERVICESession` and establish a connection to the remote service instance. If a remote service instance cannot be discovered, a new `SERVICEInstance` is instantiated locally and registered at the service discovery (via the register method of the `SERVICEOverlayCore`).

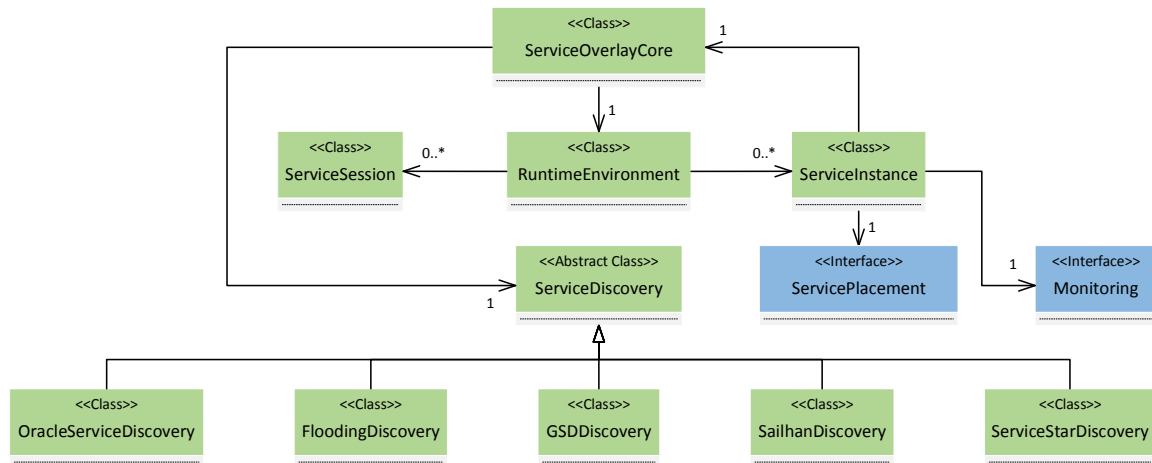


Figure 29: UML class diagram of the service discovery components

In Section 3.2.5, we identified two basic approaches for service discovery. The service discovery either sends service discovery requests through the network or it sends periodical service advertisements. We implemented a simple discovery component that provides this

behavior. The `FLOODINGDISCOVERY` can be configured with a flag to either flood service advertisements or service requests. Naive flooding-based discovery approaches are often used as baseline for performance evaluation of new service discovery approaches. We, therefore, use both approaches as baselines for our investigations, too. In both cases, the flooding radius and respective intervals and timeouts can be specified. We set the flooding radius for advertisements and service discovery requests to $\mathcal{H} = 9$ hops. Advertisements are sent in intervals of 60 seconds and they expire after 150 seconds.

As described in Section 3.2.5, we selected three different service discovery approaches to be investigated for their support for service mobility. These three discovery approaches are: the directory backbones approach by Sailhan et al. [165] (`SAILHANDISCOVERY`), the service* approach by Nedos et al. [135] (`SERVICESTARISCOVERY`), and the GSD approach by Chakraborty et al. [45] (`GSDDISCOVERY`). Details on how these approaches work can be found in Section 3.2.6.

For our simulation-based investigation, we further implemented a service discovery approach based on global knowledge. The `ORACLESERVICEISCOVERY` returns the nearest service provider upon request. It also checks whether there is a valid routing path from the service requester to the service provider. If no valid path is found, it returns a negative response. This approach constitutes the third baseline for our investigation. We discuss our findings from our investigation on service discovery in Section 5.3.

5.1.3 Service Placement

Service placement relocates service instances during runtime to ensure high quality service provision. To achieve this goal, the service placement needs information about all connected service consumers and the network to decide where to migrate the service instance to. In general, the monitoring component is required for service placement and, therefore, it is an essential part of our service overlay implementation. The classes relevant for service placement are depicted in Figure 30.

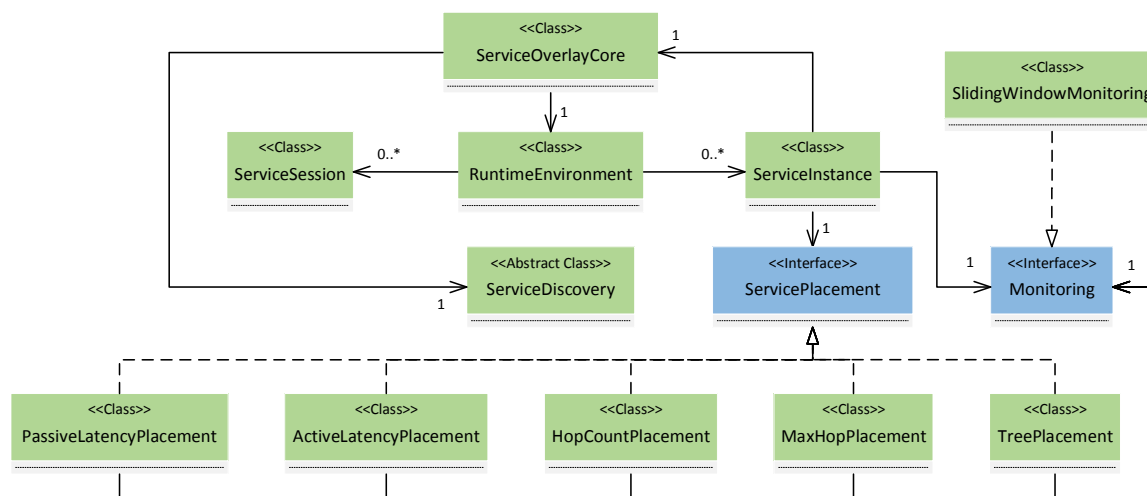


Figure 30: UML class diagram of the service placement components

Once the service instance is up and running, it produces data traffic that is sent through the network to the connected service consumers. Furthermore, service consumers also send data back to the service instance. We monitor every sent and received message at the `SERVICEINSTANCE` using the `MONITORING` interface. As a basic monitoring approach, we implemented the `SLIDINGWINDOWMONITORING` component. It can be configured to either store a fixed amount of recent messages or to store messages monitored within a fixed time span. The collected monitoring data can then be accessed by the service placement component, for instance.

Every `SERVICEINSTANCE` executes one service placement algorithm accessed through the `SERVICEPLACEMENT` interface. This algorithm is responsible to decide when a service instance should be migrated and where it should be migrated to. To migrate the service instance, the service placement algorithm calls the `MOVETo()` method of the respective `SERVICEINSTANCE` object. The service instance is then migrated to the designated destination.

As a baseline for our service placement evaluation, we implemented the tree placement approach by Oikonomou et al. [141] (`TREEPLACEMENT`) described in Section 3.2.4. This approach uses data traffic as a quality metric to decide when and where a service instance should be migrated to. In addition, we conceived and implemented four service placement approaches. Two approaches are based on the hop count metric and the other two approaches are based on the latency metric. We describe our four service placement algorithms as well as our evaluation in Section 5.2.

5.1.4 Simulation Components

To investigate service placement and service discovery approaches, we use simulations. For this purpose, we use the `PeerfactSim.KOM` simulation framework [183] (cf., Section 3.3.3). We implemented two components that model the behavior of participants to generate the load on the service overlay (cf., Section 5.1.1). The service demand generator models the intention of participants to consume various services. The service traffic generator models the actual service consumption of participants. We describe both load generators in more detail in the following sections.

Service Demand Generator

The service demand generator selects peers to act as service consumers for a specified amount of time. Each demand generator produces service requests to one specific service. Once a peer turns into a service consumer, it requests the respective service from the service discovery as described in Section 5.1.2. We implemented three different service demand generators depicted in the UML class diagram in Figure 31.

The `NoSERVICEDEMANDGENERATOR` generates no service requests at all. A group of peers that are assigned to this demand generator will never turn into service consumers. However, they might turn into service providers if the service placement component decides to migrate a service instance onto a peer of this group. The other extreme is implemented by the `COMPLETESERVICEDEMANDGENERATOR`. This demand generator turns every peer it is assigned to into a service consumer. The peer consumes the respective service for the entire duration of the simulation.

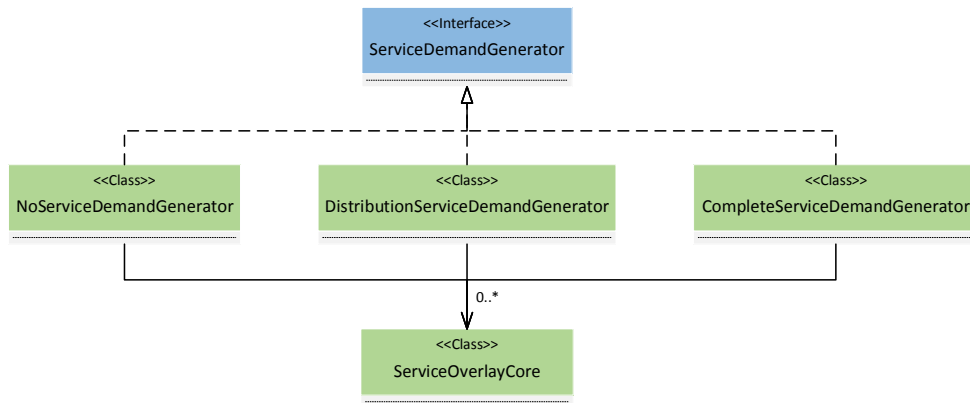


Figure 31: UML class diagram of service demand generators

The `DISTRIBUTIONSERVICEDEMANDGENERATOR` uses a distribution function to select peers as service consumers. It can be configured to use different distribution functions. Furthermore, the number of concurrent service consumers as well as the time span a peer acts as service consumer can be configured. This service demand generator then selects the appropriate number of peers to act as service consumers for the predefined time span. Once a peer is finished with service consumption, the demand generator selects another peer to be service consumer.

Service Traffic Generator

Once the `SERVICEDEMANDGENERATOR` selects a peer to become a service consumer, the peer starts interacting with the respective service. The service consumer sends data to the service and receives data back from it. An overview of all components involved in data traffic generation and data exchange is provided in Figure 32.

As discussed earlier, the `SERVICESESSION` and the `SERVICEINSTANCE` use the `SERVICECONNECTION` interface to exchange data. The `SERVICESESSION` maintains exactly one `SERVICECONNECTION` to consume the service provided by the `SERVICEINSTANCE`. The `SERVICEINSTANCE` maintains exactly one `SERVICECONNECTION` to each connected service consumer. All messages for service consumption between a service consumer and a service provider are exchanged using this network connection. Furthermore, all connection logic is handled within the implementation of the `SERVICECONNECTION` interface. We implemented the `UNRELIABLESERVICECONNECTION` to handle migration capabilities of service instances. After a service instance migrated onto another peer, the `UNRELIABLESERVICECONNECTION` component reestablishes connections automatically to all connected service consumers.

The data traffic for service consumption is generated by the `TRAFFICMODEL`. This interface is further inherited by the `CLIENTTRAFFICMODEL` and the `SERVICETRAFFICMODEL` interfaces representing the models for the service consumer and the service instance, respectively. Implemented models then generate the interaction patterns of the `SERVICESESSION` and the `SERVICEINSTANCE` and send the data using the `SERVICECONNECTION`. We implemented several traffic models to model the behavior of service consumers and service instances during service consumption. Two important models for our simulation studies are the *echo model* and the *chat model*:

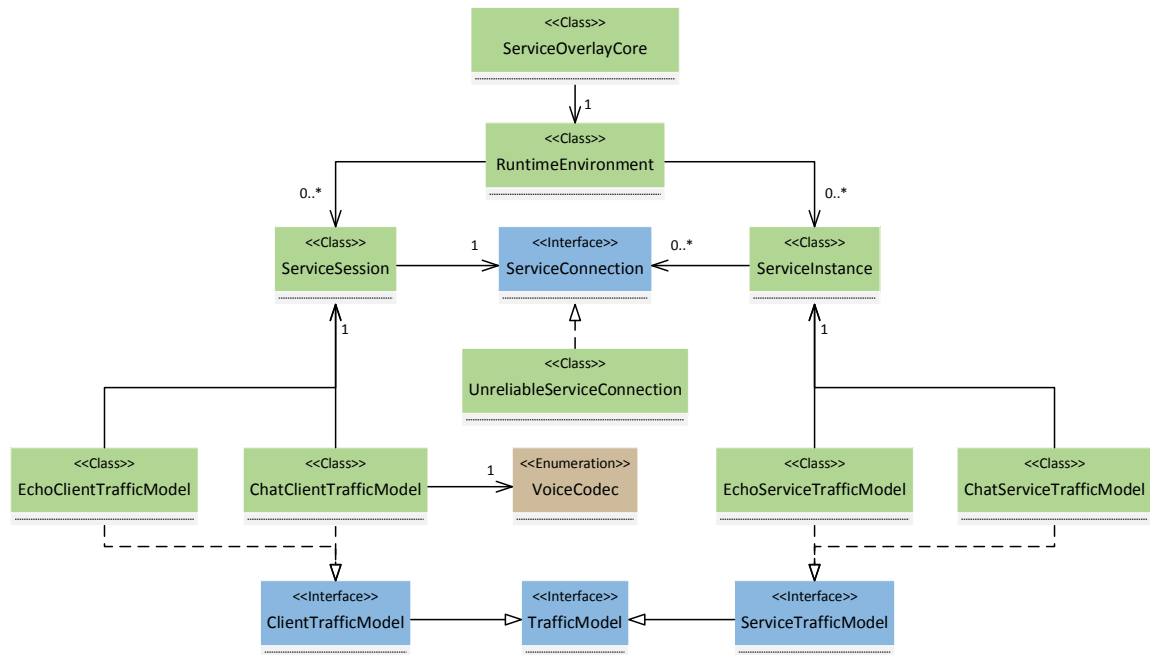


Figure 32: UML class diagram of the service traffic generators

Echo model: The echo model is a very simple traffic model. It is represented by both classes the `ECHOCLIENTTRAFFICMODEL` and the `ECHOSERVICETRAFFICMODEL`. It can be used to represent idle services, for instance. The `ECHOCLIENTTRAFFICMODEL` generates on average one message (of 100 bytes) per second to be sent from the service consumer to the service instance, e.g., keep alive messages. After receiving the message at the service instance, the `ECHOSERVICETRAFFICMODEL` directly sends an answer message of the same size back to the service consumer, e.g., acknowledgment. This model reflects a simple protocol with ongoing small message exchange between service consumer and service instance, e.g., a *ping service*.

Chat model: This model reflects voice communication between multiple service consumers connected to one service instance. It is represented by the `CHATCLIENTTRAFFICMODEL` and the `CHATSERVICETRAFFICMODEL` classes. The `CHATCLIENTTRAFFICMODEL` generates the voice packets for each service consumer. The model coordinates all service consumers to one service instance. We implemented the conversation model by Aschenbruck et al. [17] described in Section 3.3.2. This model splits the time into several conversations. During a conversation, one service consumer is picked randomly to speak for an average time of 1.6 seconds. After a small pause (0.5 seconds on average), the next random service consumer is selected to speak. With a probability of 30% the conversation ends pausing for 17.6 seconds on average before the next conversation begins.

To generate data traffic for the conversations, we implemented the codecs described in Section 3.3.2. As a standard codec, we use the G.729 codec and pack 4 frames into one voice packet. This results in one voice packet every 40 ms with a size of 40 bytes. In addition, we send one control packet per second of 16 bytes for the service instance

to recognize alive service consumers. This produces 25 voice packets and one control packet per second resulting in a data rate of roughly 1 Kbyte/s. However, UDP and IP overheads are still to be added to this value by the lower layers (cf., Section 3.3.2).

At the service provider, the service instance receives the incoming voice packets. The `CHATSERVICETRAFFICMODEL` then retransmits these packets to all connected service consumers but the source. Each incoming voice packet is only retransmitted once at the service instance via broadcast. The message is then forwarded to the service consumers using application layer multicast.

5.2 Service Placement

We identified service placement to be the most crucial component to ensure high quality service provision in our peer-to-peer service overlay concept (cf., Chapter 2 and Section 3.2.3). Service placement selects the “optimal” peer to provide a given service instance considering the current circumstances. The service instance is then migrated to the selected peer during execution. In Section 3.2.3, we identified three important optimization goals for service placement:

1. minimize data traffic generated on involved network links during service provision,
2. minimize communication delay between service consumers and service instance, and
3. minimize hop count between service instance and all service consumers.

Most state of the art approaches for service placement relocate the service instance with the goal to minimize the overall traffic load imposed on the network. We identified the tree placement approach by Oikonomou et al. [141] to be the most promising candidate that also represents other state of the art approaches with this placement goal. In thorough investigations by Wittenburg, this approach provided best results in most cases [198]. Hence, we use this service placement approach also as a baseline to evaluate new service placement algorithms designed for harsh environments.

Especially in harsh environments, we believe that minimizing communication delay and minimizing hop count will result in higher quality services. Although minimizing data traffic is a good optimization goal in general, it can lead to an unbalanced quality of service for a subset of service consumers. Under certain conditions, it can even make a service unusable to some service consumers. In our example scenario for harsh environments, disaster recovery, one important service is voice communication (cf., Chapter 2). Voice communication is sensitive to delay [56]. Especially, the mouth-to-ear delay should ideally remain below 150 ms and should not exceed 400 ms according to the ITU recommendation G.114 [93]. Therefore, we assume that a service placement strategy that uses latency as a quality metric will result in better locations for delay-sensitive services and, thus, provide higher quality services. We present our conceived latency service placement approaches in Section 5.2.1.

We investigate voice chat services that are realized as multi-point control units (cf., Section 5.1.4). Such service instances receive voice packets from one service consumer and forward them to the other connected service consumers. To decrease data traffic, the voice chat service instance uses multicast for sending the voice packets to all service consumers.

First responders form small groups to move and work in the disaster area (cf., Section 3.3). Their voice communication is organized highly hierarchical, therefore, they usually communicate with other group members and with the command center to coordinate their actions. While the group members are nearby, the command center is usually situated at the edge of the disaster area. First responders deliver their information to and get their orders from that command center. Thus, it is important that the command center as well as the first responders in the field can communicate with each other with uniform and consistent quality. We further assume that a service placement strategy using hop count as a quality metric will result in better locations for service provision in such scenarios. We present our conceived hop count service placement approaches in Section 5.2.2.

We evaluate our approaches against static placement and the tree placement approach in two different scenarios. We describe our simulation setups in Section 5.2.3. In the single group setup, a group of first responders only communicate amongst each other using a voice chat service (Section 5.2.4). In the command center setup, a static command center communicates with the first responders in the field (Section 5.2.5). Finally, we draw our conclusions on service placement in Section 5.2.6. Research on service placement was supported by the bachelor's thesis of Michael Stein [181]. A preliminary evaluation of our latency placement strategy was published in our chapter "Service Overlays" [149] in "Benchmarking Peer-to-Peer Systems" [60].

5.2.1 Latency-based Service Placement

Figure 33 depicts one shortcoming of the tree placement approach. The nodes in this figure labeled with the letter 'F' are first responders moving and working in a group. The 'C' labeled node is the command center. All green nodes are service consumers connected to the service provider (labeled 'T'). The black connections illustrate the message flow between the service consumers and the service provider 'T'. With the tree placement approach, all links at node 'T' are utilized at roughly the same rate. The service instance, thus, remains at this node although another node would provide lower latency, assuming the latency only depends on the number of hops.

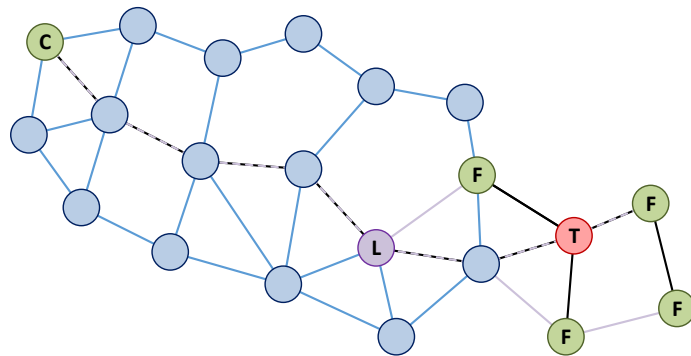


Figure 33: Placement comparison: tree placement ('T') and latency placement ('L')

Placing the service instance at the node labeled 'L' in Figure 33, the service instance should provide its service with roughly uniform latency for all connected service consumers. In this case, the purple links would be used as connections between service provider and service

consumers. Therefore, our latency placement algorithm should place the service instance on this node. To achieve this, we measure the latency at the service provider to all connected service consumers and migrate the service instance towards the service consumer with highest latency.

We conceived two different latency placement approaches. The first approach is called *active latency placement*. Active means that the service provider sends additional ping messages to all connected service consumers. From received pong messages (responses from service consumers to the ping messages), the service provider calculates the latency on each link and decides where the service instance should be migrated to. The second approach uses a passive strategy, hence, it is called *passive latency placement*. To measure the latency on each link, it uses the messages that the service instance exchanges with service consumers. This is only possible if the service instance exchanges request-response message pairs with service consumers. For instance, the voice chat service uses request-response-based control messages to recognize alive service consumers.

Active Latency Placement

The pseudo code of our active latency placement approach is presented in Algorithm 2. This approach uses epochs of 60 seconds to make a migration decision. For this purpose, the service provider sends ping messages in an interval of 25 seconds to all connected service consumers. The ping messages are not sent at once to all service consumer but distributed over time not to falsify the measurements. From the received pong messages, the round-trip-times are calculated and the last three results are stored temporarily. At the end of each epoch, the median round-trip-times are calculated for all connected service consumers. The service instance is then migrated one hop towards the service consumer with highest median round-trip-time. This should result in similar latencies for all connected service consumers. To prevent the unstable state of a service instance being continuously migrated between two peers, a migration back to the previous service providing peer is not allowed.

```
// placement algorithm for service instance SI on a service provider
// is triggered periodically every 60 seconds
begin
    // PINGCALCULATOR sends ping messages to all service consumers in
    // periods of 25 seconds and stores the resulting round-trip-times

    // retrieve peer with highest median round-trip-time from PINGCALCULATOR
    CONSUMER = PINGCALCULATOR.GETPEERWITHHIGHESTMEDIANRTT();

    // set the direct neighbor on the path to CONSUMER as migration target
    MIGRATIONTARGET = GETFIRSTHOPTOWARDSPEER(CONSUMER);

    if MIGRATIONTARGET != PREVIOUSSERVICEPROVIDER then
        SI.MIGRATETO(MIGRATIONTARGET);
```

Algorithm 2: Active latency placement

Passive Latency Placement

The pseudo code of the passive latency placement approach is presented in Algorithm 3. It does not use epochs. Instead, it measures the round-trip-times to all connected service consumers using the exchanged service messages. Therefore, no additional message overhead is generated as opposed to the active latency placement. We use a sliding window monitoring with a 30 seconds window to monitor the received and sent service messages. Every time a message is received or sent, the placement component is triggered to make a decision. At this point, the average round-trip-times for all connected service consumers are calculated and sorted in ascending order. Furthermore, the average round-trip-times over all connected service consumers and the standard deviations are calculated. The service instance is then migrated one hop towards the service consumer with highest average round-trip-time if the standard deviation is greater than 50% of the average round-trip-time. Ideally, this approach should result in a location for service provision with similar round-trip-times for all connected service consumers. The migration back to the previous service providing peer is again prohibited.

```
// placement algorithm for service instance si on a service provider
// is triggered upon receiving and sending of messages related to si
begin
    // retrieve round-trip-time values related to service instance si
    MAXRTT = MONITORING.GETMAXIMUMRTT(si);
    AVGRTT = MONITORING.GETAVGERAGERTT(si);
    STDDEVRTT = MONITORING.GETSTANDARDDEVIATIONRTT(si);

    // retrieve a list of service consumers with highest round-trip-time values
    MAXCONSUMERLIST = MONITORING.GETSERVICECONSUMERSWITHHIGHESTRTT(si);

    if AVGRTT > 0 && STDDEV / AVGRTT > 0.5 then
        for each service consumer sc in MAXCONSUMERLIST do
            // set the direct neighbor on the path to sc as migration target
            MIGRATIONTARGET = GETFIRSTHOPTOWARDSPEER(sc);

            if MIGRATIONTARGET != PREVIOUSSERVICEPROVIDER then
                si.MIGRATETo(MIGRATIONTARGET);
```

Algorithm 3: Passive latency placement

5.2.2 Hop-based Service Placement

As discussed in Section 3.3.2, the performance of VoIP communication in mesh networks is known to degrade with the number of hops [138]. We, therefore, introduce two placement strategies based on the hop count metric. Both placement strategies are passive, i.e., they only require the monitoring component to observe the usual service traffic. We assume that messages are modified by peers during the forwarding process to count all intermediate

hops. This is analog to the time-to-live field in the IP protocol. The monitoring component then just reads the hop value field from incoming messages and stores them.

Our first hop placement approach is called *hop count placement*. This strategy tries to calculate how the decision would improve or degrade service quality in terms of hop count for all connected service consumers. We call our second hop placement approach the *max hop placement*. It works analogously to the passive latency placement approach and tries to migrate the service instance towards the service consumer with highest hop count.

Hop Count Placement

The hop count placement approach is presented in Algorithm 4. It uses the sliding window monitoring to observe the 100 recent messages received at the service instance. The hop count placement algorithm is triggered whenever a new message is received at the service instance. The algorithm then calculates the average hop count for all connected service consumers and selects the service consumers with the highest average hop count. Then, for each service consumer with highest hop count, the algorithm calculates two sets. The first set contains service consumers for which the new service location will improve the hop count. The second set contains service consumers for which the new service location will not improve or even degrade the hop count.

We assume that all service consumers in the degrading set will experience a degraded service quality. Therefore, the algorithm temporarily adds one hop to the stored values of service consumers in the degraded set. Then it calculates the new maximum hop count with the incremented values. The algorithm relocates the service instance if this newly calculated maximum hop count is smaller than the current maximum hop count. Again, the service instance is only migrated if the migration target differs from the previous service provider. This helps preventing an unstable state of the algorithm. However, if no placement decision is made, the algorithm relocates the service instance if the newly calculated maximum hop count is equal to the current maximum hop count and the improving set is larger than the not improving set.

Max Hop Placement

The pseudo code of the max hop placement approach is presented in Algorithm 5. This approach again uses the sliding window monitoring to observe the 100 recent messages received from service consumers. The max hop placement algorithm is triggered whenever a new message is received at the service instance. In the first step, the algorithm calculates the average hop count for all connected service consumers. Analog to the passive latency placement approach, the max hop algorithm also sorts the service consumers according to their average hop count in ascending order. The two largest hop counts are identified for further processing.

The algorithm then checks whether there are more than one service consumers with the same amount of maximum hops. If so, the service instance is only migrated towards two service consumers that share a direct neighbor at the service provider in their respective communication path. If there is only one service consumer with maximum hop count, the service instance is migrated one hop towards that service consumer if the distance is twice as

```

// placement algorithm for service instance si on a service provider
// is triggered upon receiving and sending of messages related to si
begin
  // retrieve the maximum hop count related to service instance si
  MAXHOPS = MONITORING.GETMAXIMUMHOPCOUNT(si);

  // retrieve a list of service consumers with highest hop count value
  MAXCONSUMERLIST = MONITORING.GETSERVICECONSUMERSWITHHIGHESTHOPCOUNT(si);

  for each service consumer sc in MAXCONSUMERLIST do
    // assuming service instance would be migrated one hop towards sc
    // identify service consumers with improving and degrading service quality
    for each service consumer p in ALLSERVICECONSUMERS do
      // service consumers sharing the same direct neighbor on their path
      // will have an improved hop count after service migration
      if GETFIRSTHOPTOWARDSPEER(sc) == GETFIRSTHOPTOWARDSPEER(p) then
        | IMPROVESSET.ADD(p);
      else
        | DEGRADESET.ADD(p);
    // temporarily increment stored hop count for peers in DEGRADESET
    // and retrieve maximum hop count related to service instance si
    NEWMAXHOPS = MONITORING.GETMAXIMUMHOPCOUNTWITHDEGRADED(si, DEGRADESET);

    if (NEWMAXHOPS < MAXHOPS) ||
    (NEWMAXHOPS == MAXHOPS && LEN(IMPROVESSET) > LEN(DEGRADESET)) then
      // set the direct neighbor on the path to CONSUMER as migration target
      MIGRATIONTARGET = GETFIRSTHOPTOWARDSPEER(sc);

      if MIGRATIONTARGET != PREVIOUSSERVICEPROVIDER then
        | si.MIGRATETo(MIGRATIONTARGET);

```

Algorithm 4: Hop count placement

```

// placement algorithm for service instance si on a service provider
// is triggered upon receiving and sending of messages related to si
begin
  // retrieve hop count values related to service instance si
  MAXHOPS = MONITORING.GETMAXIMUMHOPCOUNT(si);
  MAXHOPS2ND = MONITORING.GETMAXIMUMHOPCOUNT2ND(si);

  // retrieve a list of service consumers with highest round-trip-time values
  MAXCONSUMERLIST = MONITORING.GETSERVICECONSUMERSWITHHIGHESTRTT(si);

  for each service consumer p in MAXCONSUMERLIST do
    for each service consumer q in MAXCONSUMERLIST do
      // check whether p and q share the first hop on their communication path
      if p != q && GETFIRSTHOPTOWARDSPEER(p) == GETFIRSTHOPTOWARDSPEER(q) then
        // set the direct neighbor on the path to p and q as migration target
        MIGRATIONTARGET = GETFIRSTHOPTOWARDSPEER(p);

        if MIGRATIONTARGET != PREVIOUSSERVICEPROVIDER then
          | SI.MIGRATETo(MIGRATIONTARGET);

    if LEN(MAXCONSUMERLIST) == 1 && MAXHOPS > 2 * MAXHOPS2ND then
      // set the direct neighbor on the path to service consumer with
      // highest maximum hop count as migration target
      MIGRATIONTARGET = GETFIRSTHOPTOWARDSPEER(MAXCONSUMERLIST[0]);

      if MIGRATIONTARGET != PREVIOUSSERVICEPROVIDER then
        | SI.MIGRATETo(MIGRATIONTARGET);

```

Algorithm 5: Max hop placement

far as the second largest identified hop count. To prevent an unstable state of the algorithm in both cases, the service instance is only migrated if the migration target differs from the previous service provider.

5.2.3 Evaluation Setup

To evaluate the four service placement approaches described in the previous sections, we use simulations. This allows us to create reproducible and realistic results with relatively low effort. In the present section, we describe our evaluation setup. As discussed in Section 3.2.3, we use the state of the art tree placement approach as a baseline for our evaluation. As second baseline, we use a static placement where the service instance is created on the first service consumer and is never migrated.

For simplicity, we assume homogeneous peers, hence, a matching component is not required. For the sake of reproducibility, comparability, and traceability, we use the mobility simulator BonnMotion [15] to simulate peer movement. For this purpose, we identified abstract but sufficiently realistic mobility models in Section 3.3.1. As described in Section 3.3.3, we use the peer-to-peer simulation framework PeerfactSim.KOM [183] to simulate our service overlay. It provides the required IEEE 802.11 WLAN model ported from the ns-3 simulator [184] to simulate realistic WLAN behavior and it provides necessary tools for developing new peer-to-peer systems. Table 12 summarizes all relevant network models and protocols as well as respective parameters used in our simulations.

ISO/OSI Layer	Model or Protocol	Parameters
Physical	IEEE 802.11g	rxSensitivityDbm=-96 (receive sensitivity)
		satDbm=-110 (signal attenuation threshold)
		csDbm=-99 (carrier sense)
		Dynamic bandwidth: 1 Mbit/s – 54 Mbit/s
		Frequency: 2.4 GHz
	Log-Distance Propagation Loss Model	$\gamma=3.2$ (path loss exponent)
Data Link	RTS/CTS	DSSSErrorRateModel (DSSS)
		NistErrorRateModel (ERP-OFDM)
		RTS/CTS threshold=2200 bytes
		max_retransmissions=7
		max_queue_length=400
Network	IPv4	max_time_in_queue=10 seconds
Transport	TCP	(Service migration only)
	UDP	(All remaining messages)

Table 12: Network models and parameters

We always simulate a time span of 121 minutes representing a short interval of a large rescue mission. The first minute of this time span is used as warm up phase so that network connections can be established and peers can start their movement. After the first minute

has passed, we start our measurements. As we concentrate on disaster recovery missions as an example scenario, we investigate the service placement component under these circumstances. We use a two-dimensional plane with square boundaries of $1000m \times 1000m$ to place and move mobile peers. We choose this size as it is the most commonly used plane size in simulation studies on mobile ad-hoc networks [112], thus, generating comparable results. Based on our CityMesh concept (cf., Chapter 4), we place 100 stationary nodes uniformly on this plane to interconnect them to a mesh network (10×10 mesh nodes). Incorporating the idea of cloudlets, this network is used for both message forwarding and service provision (cf., Section 3.2.2).

Interaction

We differentiate between two basic service interaction patterns. The first pattern is based on one group of mobile first responders. The group members communicate with each other using one service instance. We simulated different group sizes of $g \in [2, \dots, 10]$. In this setup, we are interested in the performance of service placement when first responders do not move in a group but alone through the disaster area. This also represents the mobility setup commonly used in service placement investigations [198]. Therefore, the movement of group members is simulated using the implementation of the Gauss-Markov mobility model (cf., Section 3.3.1) from the BonnMotion mobility simulator [15]. We use $\alpha = 0.5$ in our simulations to imitate human movement. The group members move with a slow walking speed of $1m/s$ on average to simulate movement through a harsh environment (obstructed geographical region, cf., Chapter 2).

The second pattern is based on a stationary command center and one group of mobile first responders. The group and the command center communicate with each other using one service instance. Again, we use group sizes of $g \in [2, \dots, 10]$. The command center is always placed in the top left corner and the mobile group starts at a random position on the plane. In this setup, we are interested in the performance of service placement when first responders move and work together while communicating with each other and with the command center. Therefore, the movement of the group is simulated using the implementation of the Reference Point Group Mobility model by Hong et al. [87] (cf., Section 3.3.1) from the BonnMotion mobility simulator [15]. The nodes again move with a slow walking speed of $1m/s$ on average. The maximum distance to the group center is set to 50 meters.

Routing

To also evaluate how a state of the art routing algorithm influences the performance of the service placement strategies, we use two different routing algorithms. First, we use the shortest path algorithm by Dijkstra [58] to investigate the upper bound of service placement performance without any routing overhead and with almost perfect routes. Second, we use the Ad-hoc On-Demand Distance Vector (AODV) routing protocol [157] to investigate performance of service placement strategies with realistic routing overhead (cf., Section 3.1.5).

5.2.4 Evaluation of the Single Group Setup

In the present section, we discuss the evaluation results of the first setup of our simulation study. The single group setup represents a scenario where first responders move through a disaster area while communicating with each other to coordinate their work. In this scenario, information and orders are exchanged among the group members but not with a command center. The voice communication is handled via a voice chat service. For each configuration presented in the previous section, we executed 20 simulation runs to calculate average values from the individual measurements. All plots below show the measured average values as points. Where applicable, standard deviations are shown as error bars. We connect points with lines for better illustration.

Hop Count and Mouth-to-Ear Delay

Figure 34 depicts the average hop count measured in this setup when using different group sizes. The hop count is measured between the service instance and each of the group members that receive the voice packets. When using the Dijkstra routing algorithm, all placement strategies result in an average hop count of around two hops (cf., Figure 34a). When using the AODV routing algorithm (cf., Figure 34b), the average hop count increases about one hop, especially, for groups with more than five members. Furthermore, it indicates that a static service placement on one of the group members results in the lowest average hop count. All other service placement strategies result again in very similar average hop counts.

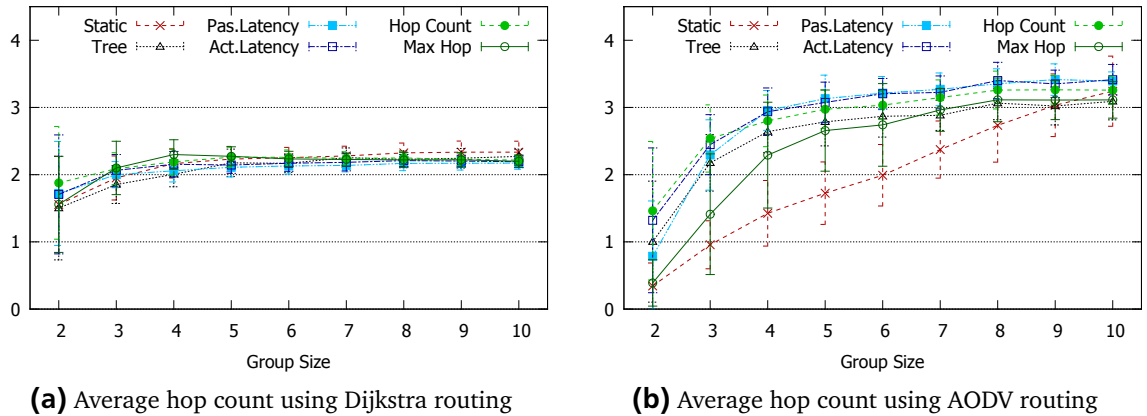


Figure 34: Average hop count for single group setup

When inspecting the mouth-to-ear delay in Figure 35 for each of the different placement strategies, we observe a similar behavior as with the average hop count. Again the static service placement seems to provide best service quality in terms of mouth-to-ear delay. Please note that the mouth-to-ear delay is always at least 40ms because of the chosen voice codec (cf., Section 5.1.4).

The performance of the static placement approach results from the placement of the service instance on one of the group members. This group member always has a hop count of 0 to the service instance and a low mouth-to-ear delay. All voice packets for this group

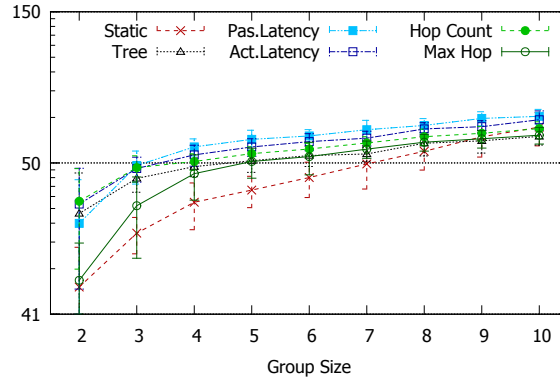


Figure 35: Average mouth-to-ear delay for single group setup using AODV routing

member are delivered locally. Other group members are further away and with this distance also comes a higher message loss probability. Lost messages are not incorporated into the average hop count values nor into the mouth-to-ear delays.

Message Loss and Service Migrations

The average message loss rate for the static placement strategy supports our assumption about this placement strategy (cf., Figure 36). Due to the dynamics in the network caused by the group movement, the communication paths between the service instance and the service consumers invalidate frequently. Before the new paths can be found by the AODV routing algorithm, the messages get lost. For the static service placement strategy the average message loss is between 30% and 40%. The average message loss rates for most of the other placement approaches are between 10% and 25%. The max hop placement approach, however, achieves similar message loss rates as the static placement. Hence, we believe this approach does not migrate the service instance very often. This is confirmed by Figure 37.

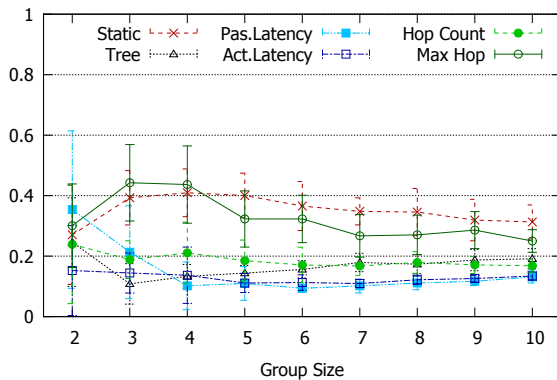


Figure 36: Message loss rates for single group setup using AODV routing

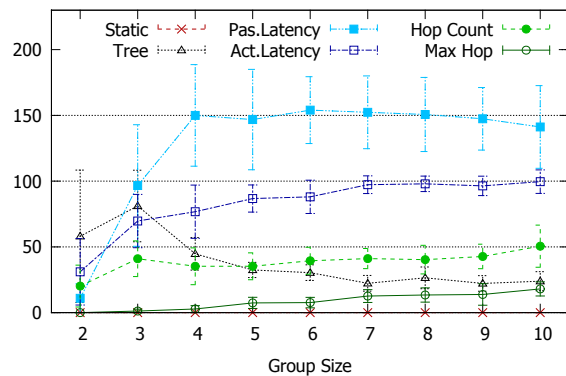


Figure 37: Number of migrations for single group setup using AODV routing

The number of service migrations indicates that the max hop placement approach is not as sensitive to changes in the network topology as the other service placement approaches (cf., Figure 37). Although the number of migrations should be kept low in general, some

amount of migrations is needed to relocate the service instance for improved service quality. The hop count placement approach, for instance, achieves good service quality in terms of mouth-to-ear delay and low message loss rates with only few migrations (less than 50 migrations in two hours). The tree placement also achieves a comparable performance.

Furthermore, this figure also shows that the passive latency placement approach migrates the service instance very frequently (more than once per minute for groups with more than three members). This results in a rather bad service quality in terms of mouth-to-ear delay due to the additional overhead caused by the numerous service migrations. However, this approach also results in very low message loss rates. For groups with more than three members, only 10-15% of the voice packets get lost.

As expected, the tree placement approach provides good performance in terms of low message loss rates, few migrations, low mouth-to-ear delays, and low hop counts. Most of the other approaches achieve comparable performance, however, the latency-based service placement approaches migrate the service instance very frequently. Furthermore, the max hop placement achieves the worst performance from our proposed service placement approaches. In fact, it performs similar to the static placement and introduces high message loss rates. At this point, we assume this service placement strategy to also fail in the command center setup.

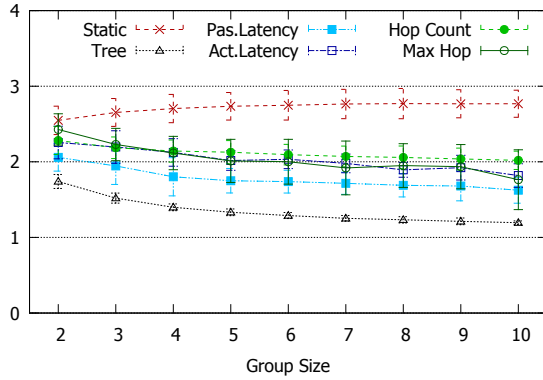
5.2.5 Evaluation of the Command Center Setup

The second setup we evaluate is the command center setup. In this scenario, a group of first responders is moving together through the disaster area. The group members exchange information and orders between each other and with a stationary command center. This command center is located at the edge of the simulated field. Hence, there is always a certain distance between the command center and the group. Voice communication is again handled via a voice chat service. For each configuration presented in Section 5.2.3, we executed 20 simulation runs to calculate average values from the individual measurements. All plots in the present section show the measured average values as points. Where applicable, standard deviations are shown as error bars. We connect points with lines for better illustration.

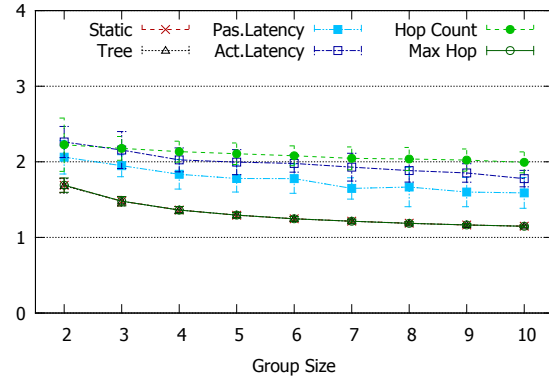
As discussed in Section 5.1.2, a service instance is created on the service consumer first trying to access the service. For the voice chat service, this means that the first participant to talk in the chat group instantiates the service instance locally. We, therefore, have to distinguish between the two possible locations for the service instance to be instantiated: one of the group members or the command center. This has consequences especially for the static service placement. Initializing the service instance on a group member then means that voice packets of the other group members will most probably have small hop counts and their communication will have small mouth-to-ear delays. However, the command center will then most probably have a worse service quality because the large distance between the group and the command center will result in a greater average hop count. It will also result in a higher mouth-to-ear delay for the communication with the command center. On the other hand, initializing the service instance on the command center will result in the opposite behavior.

Hop Count

Figure 38 depicts the average hop count measured with the Dijkstra routing algorithm and Figure 39 depicts the average hop count measured with AODV routing. Analog to the single group setup, differences in performance between the Dijkstra routing and the AODV routing algorithms become visible in these results. Again, the average hop counts are slightly higher for all service placement approaches if AODV routing is used.

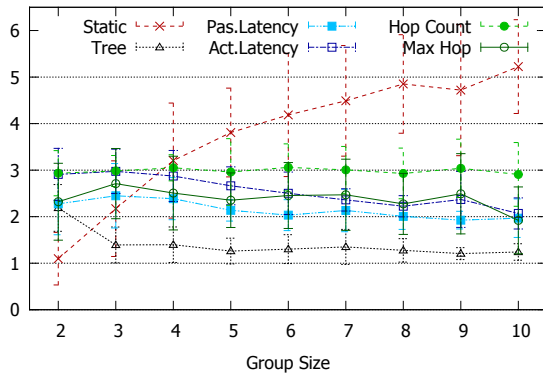


(a) Service instance initialized at command center

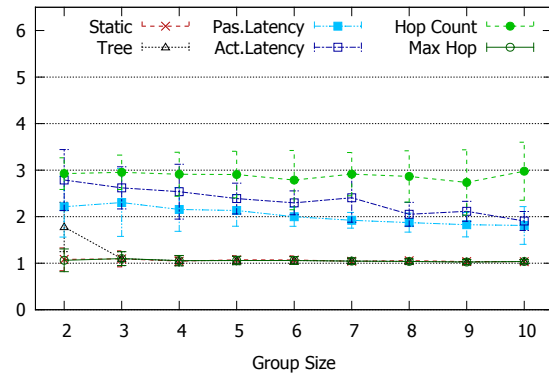


(b) Service instance initialized at group

Figure 38: Average hop count for command center setup using Dijkstra routing



(a) Service instance initialized at command center



(b) Service instance initialized at group

Figure 39: Average hop count for command center setup using AODV routing

Focusing on results measured with AODV routing, we identify the aforementioned problem. Figure 39a shows that the average hop count for the static service placement increases with the number of group members. This is because with an increasing number of group members also more voice packets must be sent from the group to the command center and back to the group. If the service instance is initialized on a group member, the average hop count for the static service placement remains slightly above one hop (cf., Figure 39b). The same applies to the tree placement and the max hop placement approaches. This might re-

sult from a lower number of migrations and higher message loss rates for these approaches, as also seen in the single group setup in the previous section.

Mouth-to-Ear Delay

Considering the service quality in terms of the average mouth-to-ear delay (cf., Figure 40), we observe a very similar behavior as with the average hop count. Again, the static service placement has a highly varying performance depending on where the service instance was initialized. Also, the max hop placement exhibits a similar behavior when initialized at a group member. This is most probably again due to the low number of service migrations and high message loss rates as seen with the single group setup. Furthermore, we can observe a rather high service quality in terms of low mouth-to-ear delay for the tree placement approach. No matter where the service instance is initialized, in both cases the tree placement approach achieves similar and high average service quality. All other approaches also perform very similar, regardless of where the service instance was initialized.

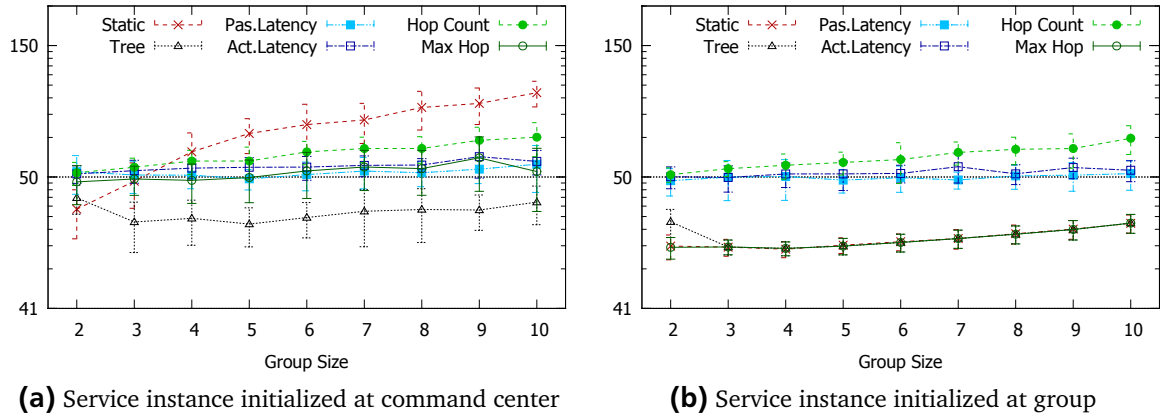


Figure 40: Average mouth-to-ear delay for command center setup using AODV routing

To further investigate the performance of the individual service placement approaches, we generate cumulative distribution functions (CDF) of both metrics the hop count and the mouth-to-ear delay. For this purpose, we pick a medium group size of six group members giving a good representation of the overall result set. The hop count CDFs in Figure 41 show the discrepancy between the static service placement on the command center and on a group member. These two distributions can be interpreted as the upper and lower bounds for service placement in a command center setup under the given constraints. The same applies to the CDFs of the mouth-to-ear delay depicted in Figure 42. All placement approaches perform inside these bounds.

Furthermore, the figures show that results for the tree placement are similar to the static service placement on one of the group members in terms of both metrics the hop count and the mouth-to-ear delay. The reason for this is that the tree placement approach tries to reduce data traffic imposed on the network. Since all service consumers produce the same amount of data, the tree placement approach places the service instance onto one of the

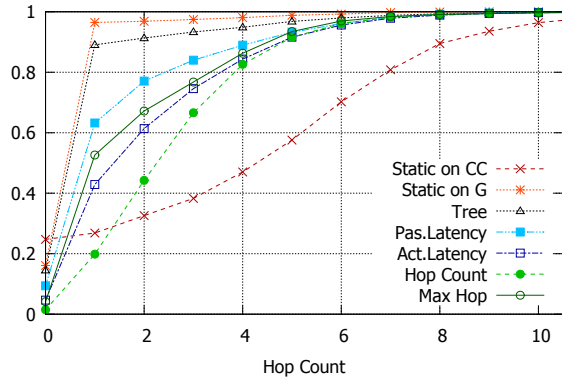


Figure 41: Hop count CDFs with 6 group members for command center setup using AODV routing

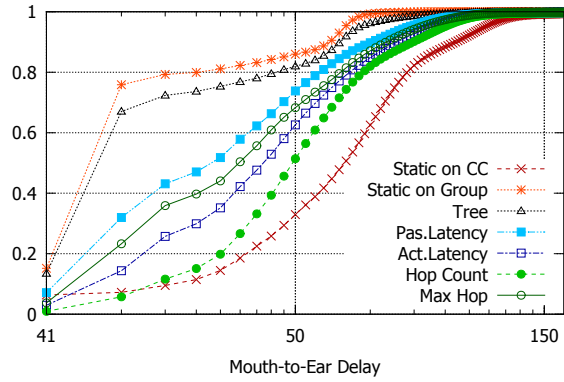


Figure 42: Mouth-to-ear delay CDFs with 6 group members for command center setup using AODV routing

group members or in the direct vicinity of the group. We discussed this behavior also in Section 5.2.1.

Per Placement Evaluation

The results indicate that the overall service quality in terms of hop count and mouth-to-ear delay varies depending on the distance of the service instance to the group members and to the command center, respectively. This implies that the service quality for the command center and for the group will differ depending on the respective distance. Therefore, we investigate both metrics for each of the service placement approaches separately. The resulting CDFs presenting the hop count metric are depicted in Figure 43. Figure 44 depicts the resulting CDFs for the mouth-to-ear delay.

Static Placement

Figure 43a shows that the static service placement on the command center results in 100% of the voice packets sent from the service instance to the command center having a hop count of 0 hops. Since the service instance is initialized on the command center peer and never migrated, this means that the voice packets are always delivered internally on that peer to reach the command center. But, the voice packets reaching the group members will always have to be sent at least one hop through the network to reach their destination. Depending on the current location of the individual group members and the communication paths discovered by the AODV routing algorithm, the distance can vary greatly in terms of hops.

On the other hand, if the service instance is initialized on one of the group members and never migrated, around 17% of all messages have a hop count of 0. We would expect this value to be roughly 14%. This value is higher than expected because the large distance to the command center introduces a high message loss probability on that communication path. As for the command center, we again observe at least one hop distance due to group movement.

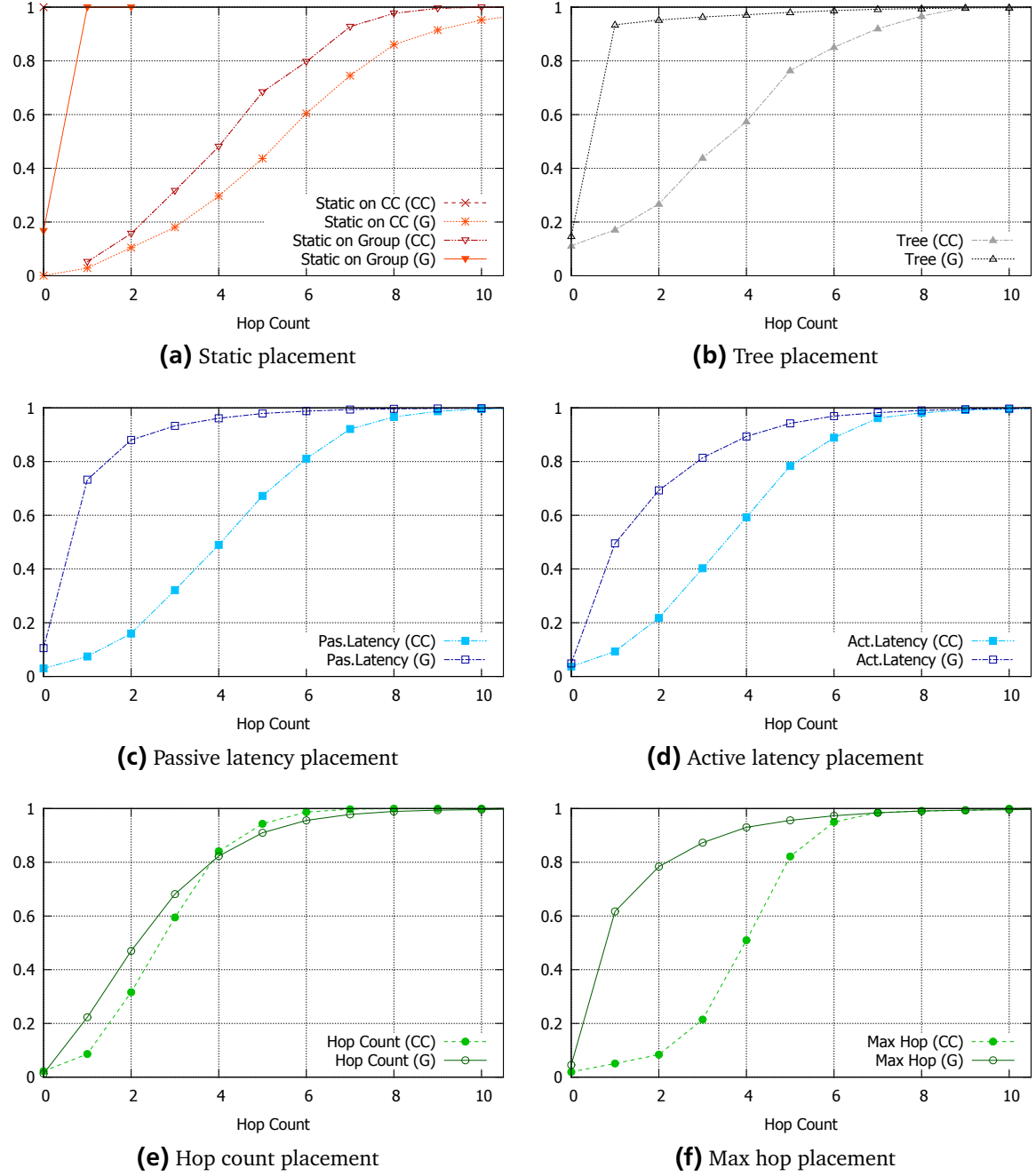


Figure 43: Hop count CDFs per placement approach with 6 group members for command center setup using AODV routing

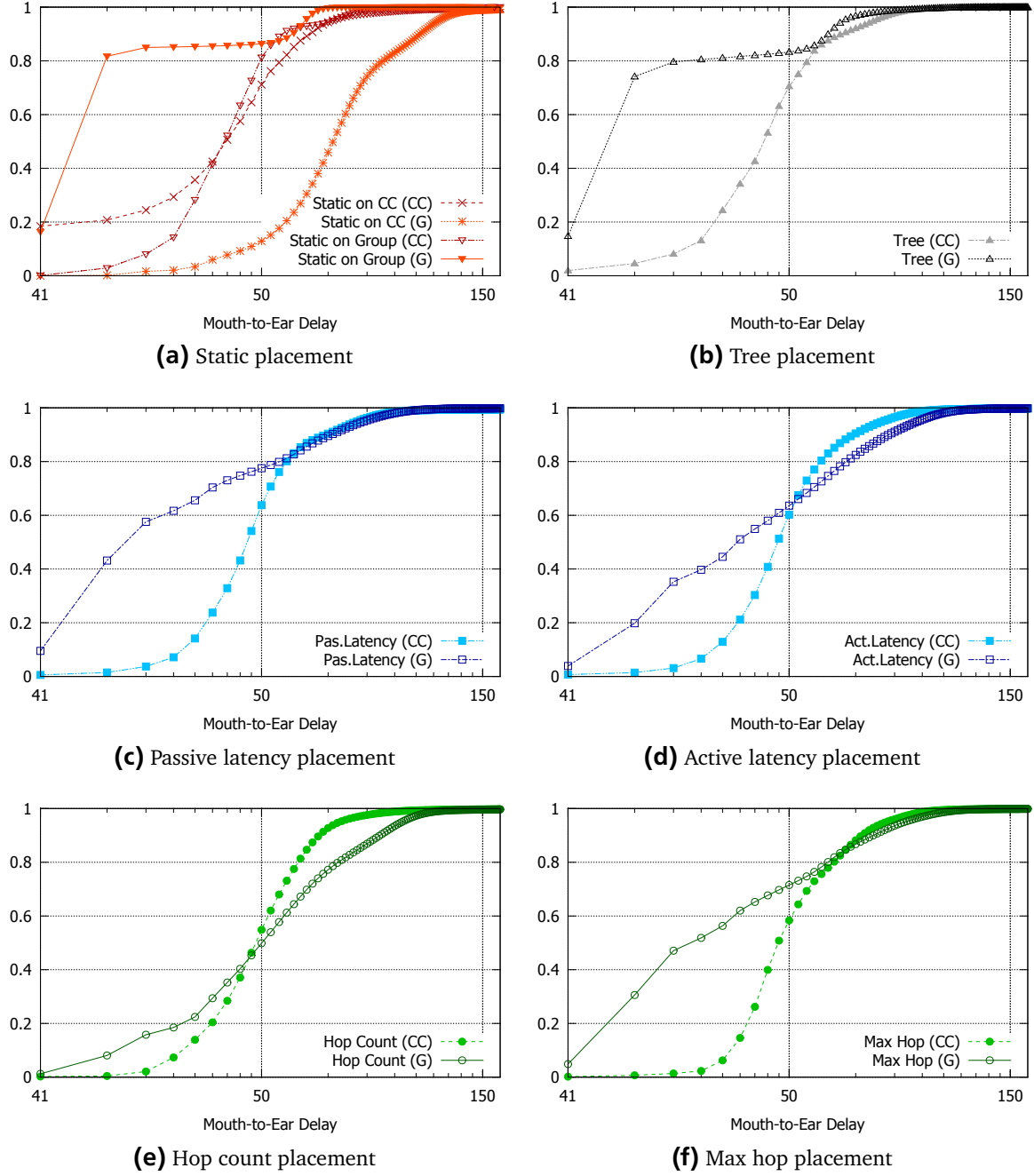


Figure 44: Mouth-to-ear delay CDFs per placement approach with 6 group members for command center setup using AODV routing

In terms of mouth-to-ear delay we observe a slightly different behavior as depicted in Figure 44a. Mouth-to-ear delay denotes the duration a packet travels from the originator of the voice packet until it reaches the receiver. This also includes the processing times introduced by the used voice codec (cf., Section 3.3.2). Therefore, this delay can be very small if the group is very close to the command center. But, it will always be greater than 40 ms.

Tree Placement

Figure 43b strengthens our assumption about the tree placement approach in a command center setup. The figure confirms our assumption regarding the discrepancy in terms of hop count between voice packets reaching the command center and those reaching group members. The results show the performance of the tree placement to be very similar to a static placement on one of the group members (cf., Figure 43a). Although the overall results indicate this approach to achieve good performance in terms of hop count and mouth-to-ear delay, the performance for the command center is rather poor compared to the performance for the group. The same holds true for the mouth-to-ear delay metric as depicted in Figure 44b. As discussed above, this behavior results from a placement in the direct vicinity of the first responder group or on one of the group members. In a command center setup, a static placement on one of the group members provides roughly the same performance as the tree placement, however, without the overhead introduced by service migration.

Passive Latency Placement and Max Hop Placement

Investigating the performance of the passive latency placement approach, the results for the command center and the group members still drift apart in terms of hop count (cf., Figure 43c) and mouth-to-ear delay (cf., Figure 44c). Also, the max hop placement cannot provide a consistent service quality for the command center and the group members. Both metrics the hop count (cf., Figure 43f) and the mouth-to-ear delay (cf., Figure 44f) drift apart as with the tree placement and the passive latency placement approaches.

Active Latency Placement

The active latency placement approach, on the other hand, seems to achieve a slightly more consistent service quality for the command center and the group members. The discrepancy is still large in terms of hop count as can be seen in Figure 43d. Only 20% of all voice packets reaching the command center have a hop count of two or less. But, for the group members around 70% of all received messages travel two hops or less through the network. The mouth-to-ear delay indicates that the service quality for the command center and the group members are quite similar as depicted in Figure 44d. For both the command center and the group members, around 60% of all voice packets have a mouth-to-ear delay between 40 and 50 ms. These results indicate that the active latency placement algorithm places the service instance between the command center and the group members. The service, thus, provides almost the same quality to the command center and to the group members. However, this comes with the price of additional message overhead introduced by the periodical ping messages.

Hop Count Placement

Finally, we investigate the performance of the hop count placement. Figure 43e shows that the discrepancy in terms of hop count can be neglected because it is very small. Around 60% of all voice packets received at the command center have a hop count of three. For the group members, less than 70% of received messages have that hop count. In terms of mouth-to-ear delay, this similarity in service quality can be confirmed as Figure 44e shows. Using the hop count placement, we achieve our goal of providing consistent service quality to all connected service consumers. The service instance is placed between all service consumers such that all service consumers can be reached with the same number of hops. It is interesting to note that by using hop count as a metric to relocate the service instance we achieve a consistent service performance in terms of both metrics the hop count and the mouth-to-ear delay.

Service Migrations and Message Loss

Figure 45 depicts the average number of service migrations for the individual service placement approaches. We see that the max hop placement and the tree placement approaches migrate the service instance far less often than the other service placement approaches. Both approaches result in a service placement in the direct vicinity of the group. Hence, the service quality for the command center is similar to the static service placement approach on one of the group members. The passive latency placement again migrates the service instance very frequently, which introduces a large overhead. In return, this influences the service quality in terms of mouth-to-ear delay negatively. We observed this behavior already with the single group setup. As seen for the hop count placement, frequent service migrations are not necessary to provide high quality services. The hop count placement achieves consistent service quality for the command center and the group members with only few migrations. The active latency placement migrates the service instance more often but does not perform as good as the hop count placement approach.

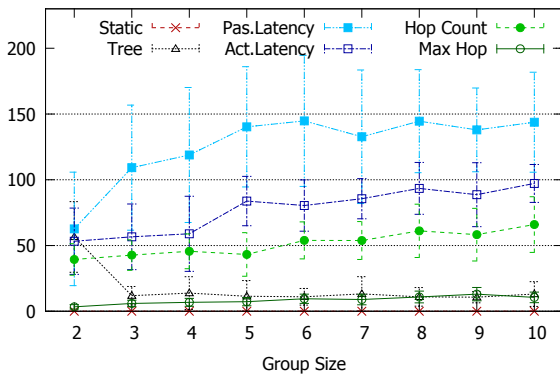


Figure 45: Average number of migrations for command center setup using AODV routing

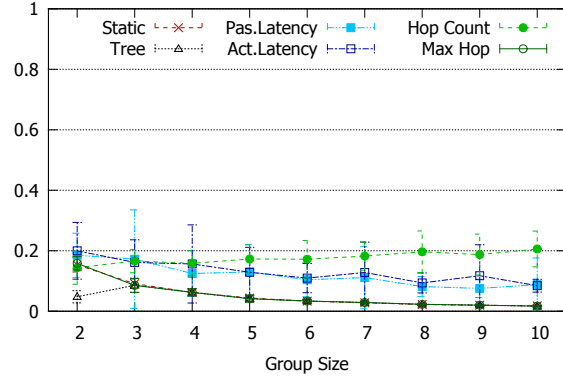


Figure 46: Average message loss rates for command center setup using AODV routing

Finally, we investigate the average message loss for all service placement approaches as depicted in Figure 46. Again, best results are achieved with the tree placement and the max hop placement approaches in terms of low message loss rates. The hop count placement

performs worst since around 20% of the voice packets get lost with any group size. Both latency-based placement approaches perform better, but still suffer from 10-20% message loss. To gain a better understanding of the individual approaches, we look at the message loss rates for the command center and the group members separately. Figure 47 shows the respective results.

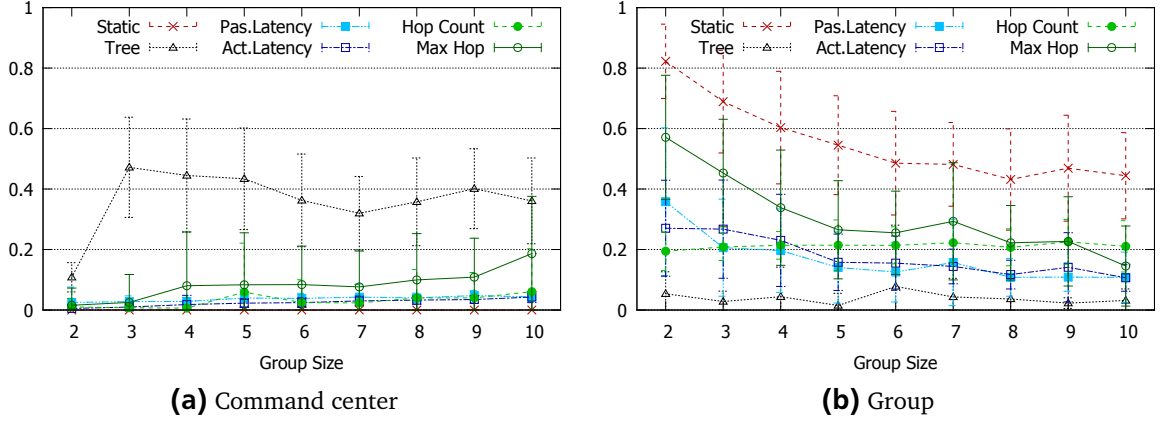


Figure 47: Message loss rates for command center setup using AODV routing

Inspecting the message loss at the command center, we clearly see that with the tree placement approach 40-50% of the voice packets never reach the command center (cf., Figure 47a). All other approaches achieve lower message loss rates at the command center. As already argued above, this is because the tree placement approach places the service instance most probably on one of the group members. On the other hand, the tree placement approach achieves lowest message loss rates for the group members. Both latency-based placement approaches lose roughly the same amount of messages and perform only slightly worse than the tree placement approach. Besides a static placement on the command center, the max hop placement has the worst performance in terms of message loss rates. The hop count placement approach achieves constant message loss rates throughout all simulated group sizes.

5.2.6 Conclusion

Our evaluation of six different approaches for service placement confirmed the good overall performance of the tree placement approach by Oikonomou et al. [141]. Especially in the single group setup, this placement approach provides good performance in terms of low message loss rates, few service migrations, low mouth-to-ear delays, and low hop counts. Most of the other approaches achieve comparable performance in the single group setup. The latency-based service placement approaches suffer from frequent service migrations. The max hop placement achieves the worst performance in terms of high message loss rates. Finally, the hop count placement achieves comparable performance as the tree placement approach in terms of low hop counts, low mouth-to-ear delays, and message loss rates.

In the command center setup, which can be found in typical disaster recovery scenarios, the tree placement did not perform as well. We demonstrated that the tree placement

performs almost as bad as a static placement on one of the first responders in the field. Overall, the tree placement approach disregards the command center in such a disaster scenario. Therefore, up to 50% of the voice packets destined for the command center never reached their destination in our evaluation. This is an issue because the command center plays a central role in disaster scenarios. Information is collected at the command center and also all orders are sent from the command center to the first responders in the field. If communication with the command center has bad quality or even fails, this can lead to inefficient mission handling and even to loss of human lives.

Also in the command center setup, the latency-based approaches again migrate the service instance very frequently. This introduces an unnecessary overhead without an improvement in service quality. Hence, these service placement approaches in their current form are not suitable for neither setup. After thorough investigation of latency-based service placement, we come to the conclusion that latency is not a suitable quality metric to base service placement decisions on. The latency on a network link over several hops is influenced by various factors and, thus, is very unstable. This effect is also known as packet delay variation and is sometimes referred to as jitter.

A more stable and also more suitable quality metric is, thus, the hop count between the service instance and all service consumers. To achieve consistent service quality for all group members as well as for the command center, we conceived the hop count placement. This service placement provides consistent service quality by minimizing the hop count for all connected service consumers. In the command center setup, the hop count placement achieves best performance in terms of consistent hop counts and mouth-to-ear delays for the command center and all group members. Furthermore, it achieves this goal with only 20 to 30 service migrations per hour and moderate overall message loss of up to 20%.

5.3 Service Discovery

In the previous section, we designed four service placement strategies based on hop count and latency as quality metrics. We evaluated these placement approaches in two disaster recovery setups using the state of the art tree placement approach as a baseline. In the present section, we evaluate the performance of service discovery in our peer-to-peer service overlay concept for harsh environments. Our goal is to identify a suitable service discovery approach for harsh environments capable of handling peer and service mobility.

In Section 3.2.6, we selected three service discovery candidates for our evaluation. We selected the Directory Backbones approach proposed by Sailhan et al. [165] to represent directory-based approaches. To represent directory-less approaches, we selected the Group-based Service Discovery (GSD) by Chakraborty et al. [45] and the Service* approach by Nedos et al. [135]. In our evaluation, we call these approaches *Sailhan*, *GSD*, and *Service**, respectively. As baseline for our evaluation, we use three additional service discovery approaches as described in Section 5.1.2. Two approaches commonly used as baseline for performance evaluations of service discovery are based on message flooding. The *Flood_{ADV}* approach periodically floods advertisements from service instances through the network. And the *Flood_{REQ}* floods service discovery requests through the network. The third discovery baseline is based on global knowledge. Upon request, it returns the nearest service provider and checks for a valid routing path from service requester to service provider. We call this approach *Oracle*.

We describe our evaluation setup in Section 5.3.1. First, we investigate the discovery performance of the selected approaches considering peer mobility and static service placement. For this purpose, we measure the data traffic generated by the individual discovery approaches and present the results in Section 5.3.2. As discussed earlier, we believe that service placement has an impact on the performance of service discovery. Therefore, we investigate the performance of service discovery considering service mobility caused by service placement and present the results in Section 5.3.3. Finally, we draw our conclusions on service discovery in Section 5.3.4. Research on service discovery was supported by the master’s thesis of Tobias Bönning [32].

5.3.1 Evaluation Setup

For our evaluation of the service discovery approaches, we use an evaluation setup analog to the service placement evaluation setup (cf., Section 5.2.3). For the sake of comparability, traceability, and reproducibility, we use the mobility simulator BonnMotion [15] to simulate peer movement. Furthermore, we use the peer-to-peer simulation framework PeerfactSim.KOM [183] to simulate network communication between peers as it provides the required realistic IEEE 802.11 WLAN model ported from the ns-3 simulator [184]. We also use the same simulator configuration, models, protocols, and respective parameters for our discovery evaluation as presented in Table 12. For routing, we again use both routing protocols the Ad-hoc On-Demand Distance Vector (AODV) routing [157] and the shortest path algorithm by Dijkstra [58]. In our simulations, we use one minute as warm up phase and afterward we simulate a time span of 120 minutes to conduct our measurements. This represents a short interval of a large rescue mission.

We use a two-dimensional plane with square boundaries of $500m \times 500m$ to place and move mobile peers. We choose this size as it represents a medium sized disaster area including incident site, treatment and transport zones as described by Aschenbruck et al. [16]. On that plane, we place $p \in [50, 100, 150, 200, 250]$ mobile peers to investigate the ability of service discovery approaches to scale with the number of peers (horizontal scaling). We simulate peer mobility using the Gauss-Markov mobility model from the BonnMotion framework to generate comparable and reproducible results (cf., Section 3.3.1).

Service Interaction

Besides investigating horizontal scaling, we also investigate the impact of an increased number of services (vertical scaling). This allows us to investigate the emerging data traffic and the performance of the discovery approaches along both scaling dimensions. We use $s \in [1, 3, 6, 10, 15, 21, 28, 36, 45]$ different service types in our simulations to investigate vertical scaling effects. Each service type can be interpreted as a voice chat service that enables a small group of recovery workers to communicate with each other. To keep our measurements sensitive to the emerging discovery traffic, we use the *echo model* traffic pattern described in Section 5.1.4. This traffic pattern then represents an idle voice chat group that only exchanges keep alive messages between service consumers and the service provider. No actual voice communication is generated by this traffic pattern.

For each service type, five random service consumers are chosen on average to interact with the service instance at the same time. To generate dynamic service discovery requests,

the service consumption of service consumers is limited to 10 minutes on average. After that period, another peer is chosen randomly to consume the service.

5.3.2 Evaluation of Discovery Performance with Static Service Placement

In the present section, we discuss the results of our simulation study based on static service placement. For each configuration presented above, we executed 20 simulation runs to calculate average values from the individual measurements. All plots below show the measured average values as points or as bars. Where applicable, standard deviations are shown as error bars. We connect points with lines for better illustration. First, we investigate the data traffic imposed on the transport layer when using the individual service discovery approaches. Second, we evaluate the performance of the selected service discovery approaches in terms of successful service requests.

In direct comparison, the results obtained with the shortest path algorithm by Dijkstra always have smaller data traffic and better discovery performance compared to the results obtained with the AODV routing algorithm. However, both results are very similar in their overall statement. The shortest path algorithm is based on global knowledge, which is not available in real setups. Therefore, we focus on results obtained with the AODV routing protocol in our following evaluation.

Data Traffic

To investigate the additional data overhead imposed on the system by the selected discovery approaches, we measure the generated data traffic on the transport layer. The discovery data traffic is not the only kind of traffic generated on the transport layer. As described before, service consumption is also handled through this layer. Also, service migration and other management overhead for the peer-to-peer service overlay is generated on the transport layer. Hence, we classify the traffic into discovery-related data traffic, service consumption traffic, and management traffic. Management traffic is very small in most cases and can be neglected.

Figure 48 shows our results obtained with 50 peers. In Figure 48a, we see that all discovery approaches generate very similar amounts of data traffic. The Flood_{ADV}, GSD, and Service* approaches, however, generate more data traffic compared to the other approaches. Based on a configuration with 45 service types, the Oracle discovery, Flood_{REQ}, and the Sailhan discovery generate roughly 250 Mbytes during a time span of 120 minutes. The Flood_{ADV} approach generates twice as much data traffic.

To understand the reasons for this discrepancy, we analyze the distribution of the different traffic types depicted in Figure 48b. In this figure, we see how the imposed data traffic increases with the number of different service types in the system if using the Flood_{ADV} and the GSD approaches. Both approaches send periodic advertisements for each of the running service instances. Hence, the discovery-related data traffic increases heavily with the amount of service types as more advertisements are sent.

The data traffic generated by the service instances is nearly identical for all service discovery approaches. The service discovery traffic generated by individual service discovery approaches is, hence, solely responsible for differences between the approaches. Furthermore, the Oracle approach does not impose any discovery-related data traffic. For this

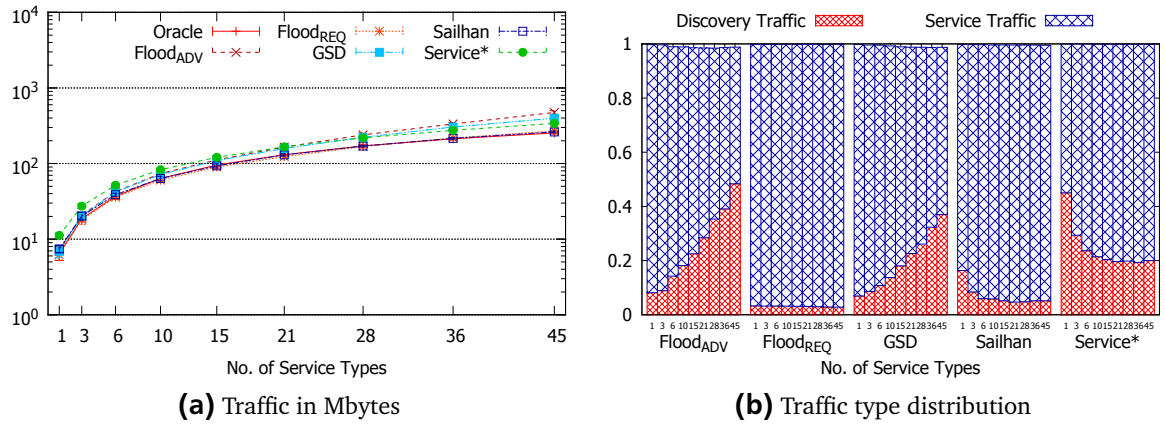


Figure 48: Data traffic on the transport layer generated by different discovery approaches with static service placement and 50 peers

reason, we do not depict this approach in Figure 48b and related figures. As mentioned above, the management traffic can be neglected due to its small amount. In Figure 48b, the management traffic accounts for less than 3% when using the GSD approach, for instance.

Based on simulations with 50 peers, the Service* and the Sailhan approaches seem not to be affected negatively when increasing the number of service types. In fact, these approaches generate a more or less constant data traffic related to service discovery. Both approaches seem to require a small number of running services to be profitable. This is because both discovery approaches work very efficiently in such small networks (50 peers). However, the most efficient service discovery is Flood_{REQ}. Compared to the generated service traffic, the Flood_{REQ} discovery generates only a small amount of the overall data traffic.

Results obtained with 150 peers (cf., Figure 49) and 250 peers (cf., Figure 50) demonstrate the drastic influence of the increasing number of peers on the efficiency of the selected discovery approaches. Again, Flood_{ADV}, GSD, and Service* impose more data traffic on the system than the other service discovery approaches (cf., Figure 49a and Figure 50a). It is interesting to note that for these network sizes the Sailhan approach imposes less overall data traffic on the system than any other discovery approach. The reason for this behavior is the bad discovery performance of this approach. The Sailhan approach can only discover a small fraction of service instances successfully. The service-related traffic as well as the overall data traffic are, thus, very small for this approach. We elaborate on this fact in the following section.

Figure 49b and Figure 50b again depict the distribution of the two traffic types. We observe the raise of discovery traffic with increasing number of service types for the Flood_{ADV} and the GSD approaches, and a generally high discovery traffic rate for the Sailhan and the Service* approaches. Analog to the small network size of 50 peers, the discovery-related traffic for the Flood_{ADV} and the GSD approaches increases heavily with both the number of service types and the number of peers. The Sailhan approach generates a high amount of discovery-related data traffic. However, service instances cannot be discovered successfully. Thus, service instances do not account for much data traffic imposed on the entire system. The Service* approach also imposes a huge amount of discovery-related data traffic on the system, however, not as bad as the Sailhan approach.

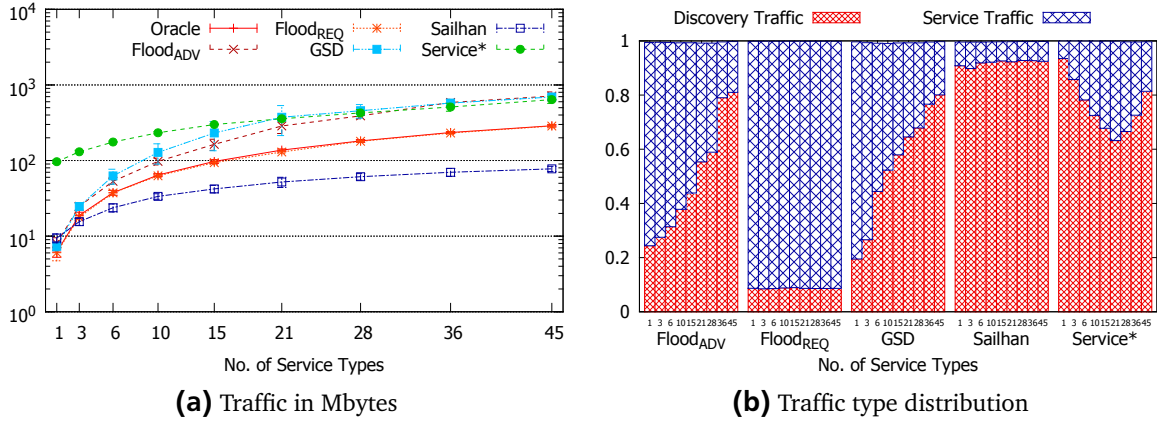


Figure 49: Data traffic on the transport layer generated by different discovery approaches with static service placement and 150 peers

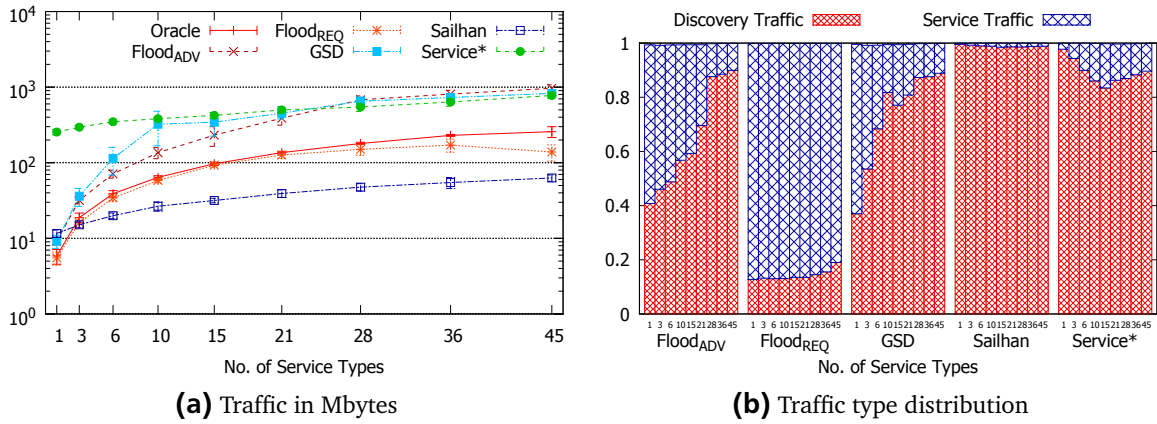


Figure 50: Data traffic on the transport layer generated by different discovery approaches with static service placement and 250 peers

In general, we believe that these high traffic rates result from the high peer density on the relatively small field. The selected discovery approaches can handle sparse networks but fail in dense networks. The approaches introduce a high amount of data traffic, a large fraction being caused by discovery-related traffic. Surprisingly, the Flood_{REQ} approach generates very low overall data traffic under all considered constraints. Furthermore, the discovery-related data traffic increases slower in consideration of both scaling dimensions than with any other discovery approach. In the next section, we investigate the discovery performance in terms of successful service discovery requests.

Discovery Performance

After investigating the data traffic generated by the selected service discovery approaches, we evaluate the discovery performance of these approaches in the present section. Again, we use static service placement for all service instances in the system. We measure the

performance of the discovery approaches in terms of positive requests and true positive requests. Rates for both metrics are calculated against the number of started service requests. Positive requests are requests that returned a positive answer, indicating the corresponding service instance was found. However, only true positive requests lead to successful service connections. False positive requests lead to a connection timeout or to connection refusal because the service requester tries to contact a peer for a connection to a service instance that the peer does not host (anymore).

Figure 51 shows our results obtained with 50, 150, and 250 peers. The first thing to notice is that the Oracle approach does not achieve 100% positive requests. As described before, this discovery approach checks for a valid routing path from the service requester to the requested service instance. If there is no valid path or the service instance is not yet being executed, the request is answered negatively. However, every positive answer always results in a successfully established connection to the requested service instance. Hence, the rates of the true positive requests are always identical to the rates of positive requests with the Oracle discovery.

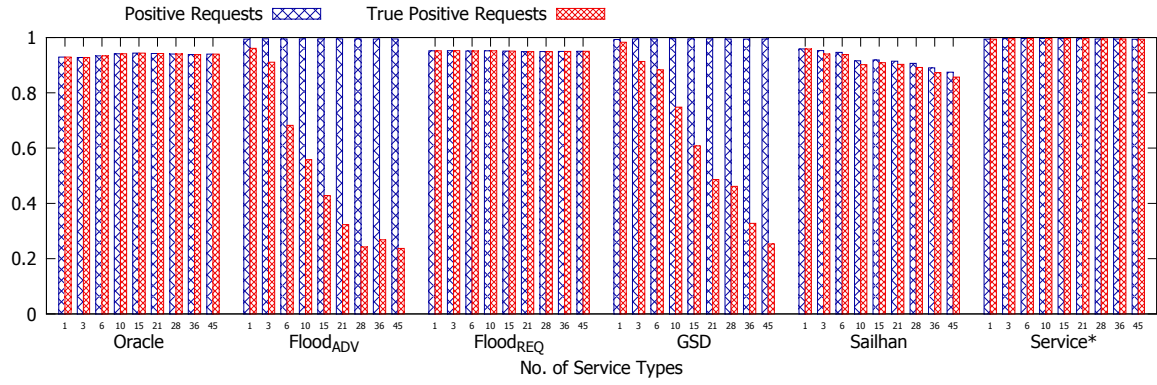
The Flood_{REQ} approach achieves nearly the same discovery performance as the Oracle approach in networks with 50 (cf., Figure 51a) and 150 peers (cf., Figure 51b). In networks with high peer density, however, the performance decreases. Especially, a high number of different service types in the network has a negative influence on the discovery performance (cf., Figure 51c). This is most probably due to the AODV routing algorithm. Network topology changes regularly in highly dense and dynamic networks. Therefore, the routes invalidate frequently and responses of service providers do not reach the service requesters before the requests time out.

Analog to the data traffic results from the previous section, the Flood_{ADV} and the GSD approaches again show a similar behavior in terms of discovery performance. Discovery requests are handled using the advertisements received earlier. By the time requesting a service, these advertisements might not be valid anymore. Furthermore, we see that the Sailhan approach performs very badly in dense networks (cf., Figure 51b and Figure 51c). These very low rates of positive requests are the reason why the data traffic drops so deep for this approach compared to the others. Although we believe this is due to the inability of the Sailhan approach to cope with highly dense and dynamic networks, we cannot eliminate the possibility of implementation errors.

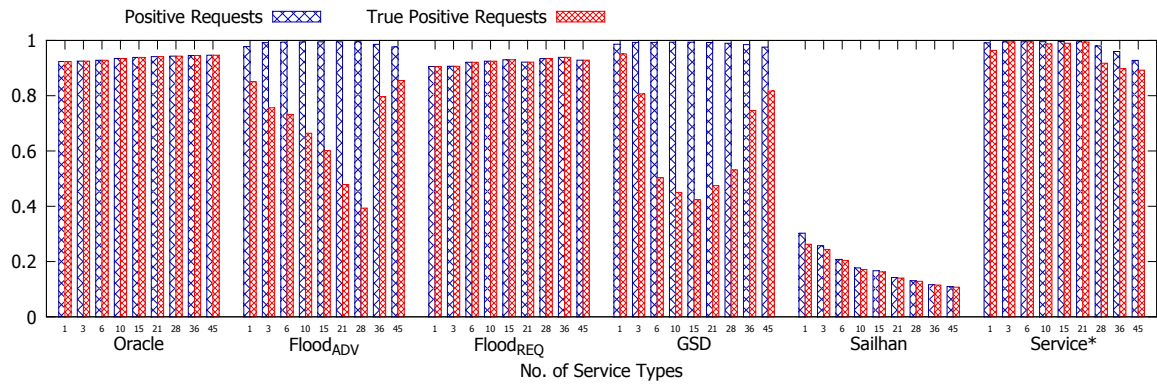
The most surprising fact, however, is the performance of the Service* approach. In sparse networks, it achieves nearly 100% true positive requests (cf., Figure 51a). Furthermore, discovery performance is stable in both scaling dimensions. Even in highly dense networks, more than 80% of the started requests lead to a successful service connection (cf., Figure 51c). However, this good performance comes with a high amount of discovery-related data traffic imposed on the system we saw in the previous section. In the next section, we investigate the influence of service placement on the performance of the selected service discovery approaches.

5.3.3 Evaluation of Discovery Performance with Service Mobility

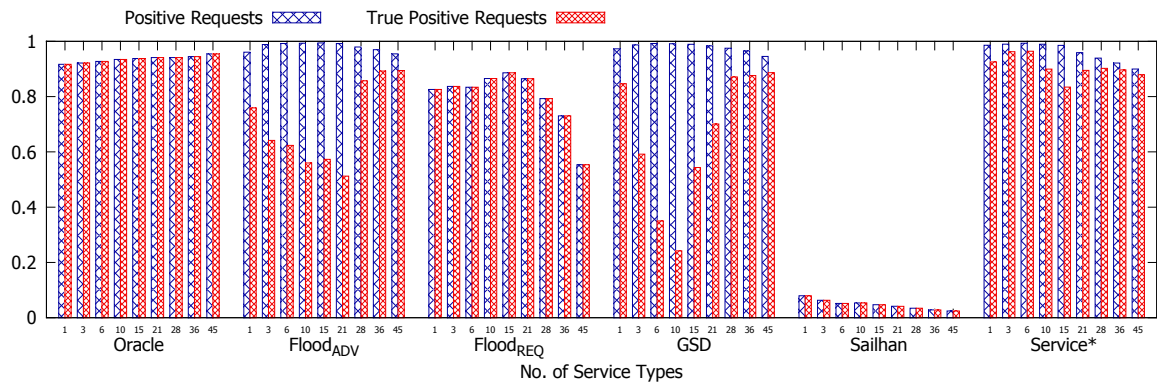
In the present section, we investigate how the service placement component influences the performance of the individual discovery approaches. The service placement component migrates the service instance onto other peers to increase the service quality for connected



(a) 50 peers



(b) 150 peers



(c) 250 peers

Figure 51: Performance of different discovery approaches with static service placement and a varying number of peers

service consumers. A selected service discovery candidate must be able to handle these additional and dynamic changes. Otherwise, it is impractical to be used as the service discovery component of a peer-to-peer service overlay for harsh environments.

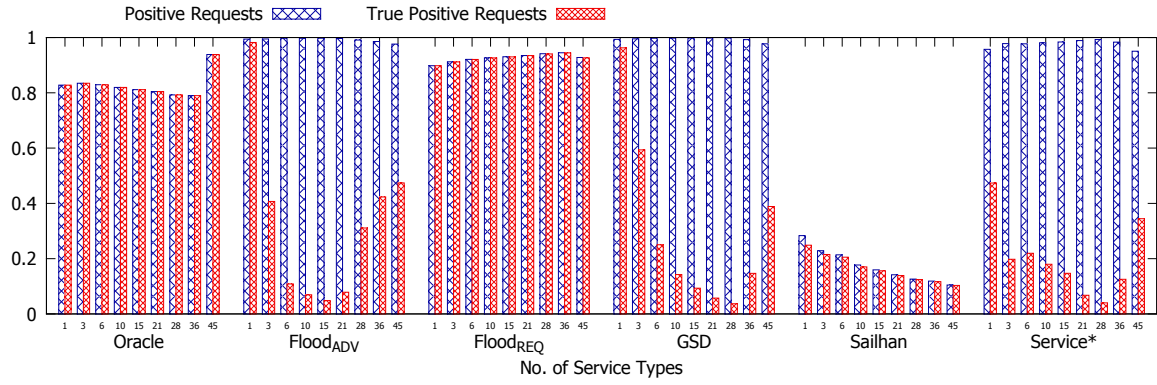
For our evaluation, we selected the hop count placement, the active latency placement, and the tree placement approaches investigated in Section 5.2. Our evaluation showed the hop count placement to be the best fit for the service placement component in harsh environments. We use the active latency placement approach to introduce high dynamics through service mobility as it migrates the service instance frequently. Finally, we select the tree placement as it is the state of the art service placement approach.

We choose a medium network size of 150 peers. This evaluation setup is comparable with the single group setup we used during the service placement evaluation (cf., Section 5.2.4). We use a group size of five peers for our evaluation, thus, the number of migrations correlates with our results depicted in Figure 37. With five service consumers, the hop count placement and the tree placement approaches migrate a service instance with nearly the same frequency. The active latency placement, however, migrates a service instance roughly twice as often as the other two approaches. These facts are important for the interpretation of our results depicted in Figure 52.

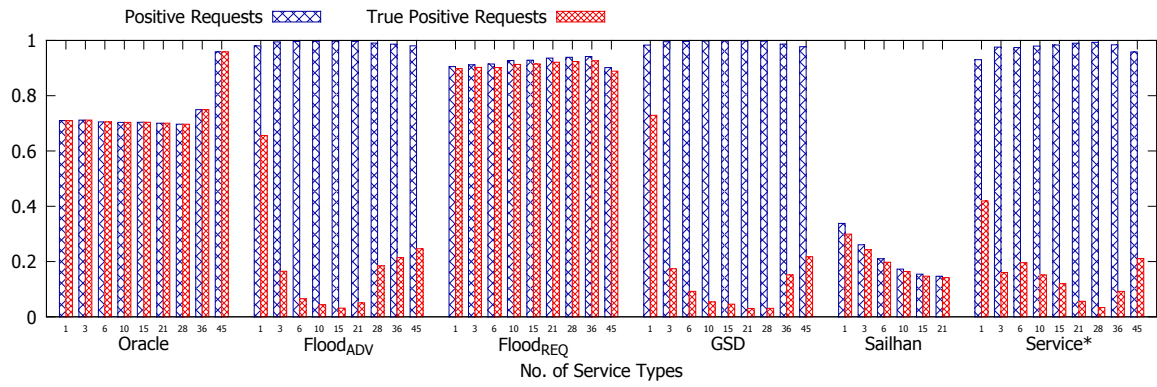
Figure 52a depicts our results for the hop count placement. We observe a decrease in positive requests for the Oracle approach compared to the results above with the static service placement. The results for the tree placement approach in Figure 52c are very similar due to the similar migration frequency. As the active latency placement approach has a higher migration frequency, we observe a further decrease in the positive request rates for the Oracle discovery approach (cf., Figure 52b). However, still more than 70% of the requests lead to a successfully established service connection. This is due to the checking for valid routing paths of the Oracle discovery. Frequent migrations increase the probability that the AODV routing algorithm at the service requester has not yet discovered a route to the current service provider. Thus, the Oracle discovery responds negatively to the discovery request.

As indicated by Figure 52, the Flood_{ADV}, the GSD, and the Service* approaches can all achieve almost 100% positive requests with any service placement approach. However, the information retrieved by these discovery approaches is outdated and the true positive request rates drop below 40% in most cases. With higher service migration frequency, the true positive request rates decrease further for all discovery approaches. All these approaches can handle the discovery of only one service type quite well and can achieve up to 98% true positive requests. But, increasing the number of service types decreases the performance drastically. Since all these approaches are based on some kind of advertisement flooding, the information stored in the local caches at service requesters is often outdated. This then results in the low true positive request rates.

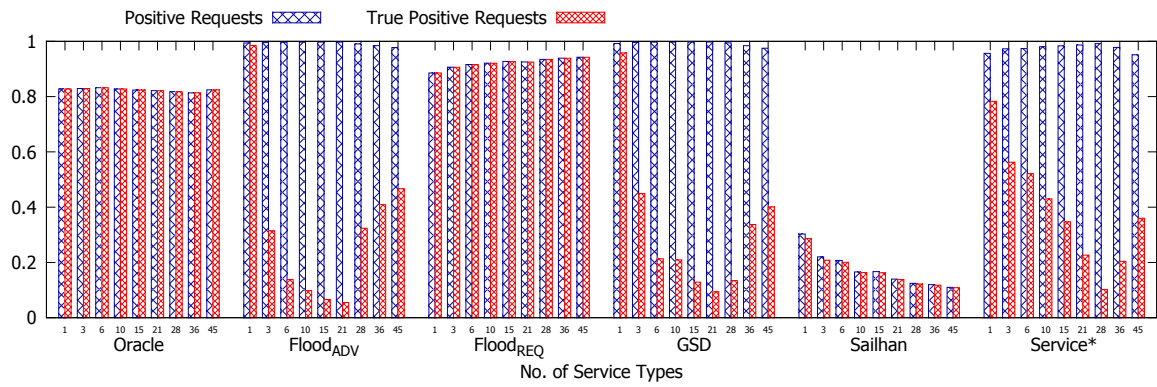
After interpreting the results depicted in Figure 51, it is not surprising that the Sailhan discovery again shows a bad discovery performance. The discrepancy between positive requests and true positive requests, however, is not as bad as compared to the three other discovery approaches described above. If a request is answered positively, it leads to a successful service connection in almost all cases. The service directories of this approach hold correct information about service instances but the high network density interferes with the successful distribution of that information. Advertisements are not properly propagated between the directory backbones.



(a) Hop count placement



(b) Active latency placement



(c) Tree placement

Figure 52: Performance of discovery approaches with different service placement strategies and 150 peers

Finally, we again focus on the Flood_{REQ} discovery approach. We observe a very high discovery performance of more than 80% true positive requests for all cases. At first glance, it seems surprising that this approach achieves higher rates than the Oracle discovery. But, as explained before, the Oracle discovery answers service requests positively only if there is a valid routing path to the requested service instance. This means that the routing path must be already established in the AODV routing tables of the respective peers. However, AODV routes are, as stated by the name, discovered on demand. If the service requester did not try to contact the current service provider before, the route to this peer is missing. With the Flood_{REQ} approach the routing path is established once the service provider responds to the received request. Hence, this approach can achieve higher rates than the Oracle approach when an on-demand routing protocol is used.

5.3.4 Conclusion

Our evaluation of six different service discovery approaches revealed surprising results. We expected service discovery approaches from literature to achieve good service discovery performance in terms of positive request rates and low discovery-related data traffic. Especially because all selected service discovery approaches were designed to be used in mobile ad-hoc network environments. However, we found that the Group-based Service Discovery (GSD) by Chakraborty et al. [45] performs very similar to a flooding approach based on periodic advertisement flooding, which we also used as baseline. Both approaches impose similar discovery-related data traffic on the network and achieve a comparable performance in terms of true positive discovery requests.

Furthermore, the Service* approach by Nedos et al. [135] achieves a rather high discovery performance in terms of true positive discovery requests. However, this comes at the price of very high discovery-related data traffic imposed on the network. High data traffic should be avoided so that more bandwidth is available for service provision. Therefore, this approach also does not qualify for service discovery in harsh environments.

The Directory Backbones approach proposed by Sailhan et al. [165] is quite efficient in small or sparse networks. However, with an increasing number of peers, this approach also fails and results in very bad discovery performance. Service instances cannot be successfully discovered in dense networks. And with additional service mobility, the discovery performance drops even further. This makes the directory backbones approach unusable for harsh environments.

The most surprising fact we discovered during our evaluation is the performance and efficiency of the naive request flooding approach. The peer movement and also the frequent migrations of service instances were not able to have a drastic negative impact on the discovery performance of this simple approach. Data traffic generated by this discovery approach was smallest from all investigated discovery approaches and under all investigated constraints. Discovery performance was very high and only dropped slightly in very dense networks and with a high number of service types. Therefore, we recommend the usage of this simple request flooding approach for service discovery in peer-to-peer service overlays for harsh environments.

5.4 Smooth Service Migration

In the previous sections, we described the system model of our proposed peer-to-peer service overlay (cf., Section 5.1). We investigated several service placement approaches and identified our proposed hop count placement to perform best in command center setups often found in disaster scenarios (cf., Section 5.2). Hop count placement is able to provide consistent service quality to all connected service consumers with only few service migrations. We further investigated selected state of the art service discovery approaches for their support for service mobility introduced by service placement (cf., Section 5.3). We showed that a naive approach based on request flooding performs best in terms of discovery performance and low discovery-related data traffic in scenarios with peer and service mobility.

As described in Section 3.2.1, service migration is the concept that enables service mobility and is, thus, required for service placement. During service migration, the service is usually interrupted and not available to service consumers for a short time (e.g., voice communication provided by a voice chat service would stutter or interrupt during migration). To prevent this from happening, we propose several concepts for continuous service provision through smooth service migration. These concepts include soft handover, persistent socket connections, efficient serialization, and an efficient network protocol. In the present section, we first introduce our real-world experiment setup to investigate smooth service migration (Section 5.4.1). Afterward, we present the graphical user interface of our service overlay prototype in Section 5.4.2. Finally, we present the four concepts for smooth service migration in Section 5.4.3.

5.4.1 Real Experiment Setup

To investigate service migration and possible interruptions during service provision, simulations are not suitable and a real experiment setup is required. Especially when we recall our example service (voice chat), interruptions during voice communication would be hard to measure in a simulation setup. We, therefore, implemented a prototype of our peer-to-peer service overlay on top of a mobile operating system. Furthermore, we used this prototype to implement and investigate our concepts for smooth service migration.

Until today, many different mobile operating systems, such as Android, iOS, or Windows Phone, provide a rich platform for app development. At first glance, each of those platforms could be used to implement a peer-to-peer service overlay prototype. However, most of them have strong restrictions, which complicate the technical realization. To focus on the implementation of our service overlay framework, these restrictions must be reduced. For this purpose, we selected the Android platform by Google Inc. to implement our peer-to-peer service overlay prototype. Besides its open source nature, the Android platform has the benefit of being based on the Java language. This has the advantages that implemented Android apps run on a Dalvik virtual machine, which already realizes the runtime environment described in Section 2.3. The existence of this virtual machine also implicates more or less homogenous peers in terms of software resources.

The Android operating system also provides easy to use functionality to access the Internet via the built-in WiFi adapters of Android-based devices. Unfortunately, this does not mean that ad-hoc functionality is easily accessible, too. In fact, Google Inc. made several

changes to the Linux and Java foundation of Android and disabled the ad-hoc functionality. Furthermore, the IEEE 802.11s standard [86], which is part of the Linux kernel, is missing on Android. Therefore, it is not possible to create WiFi ad-hoc or mesh networks with a stock Android device. This is an issue because mesh networks constitute the basis for our proposed peer-to-peer service overlay. We, therefore, first describe how to reactivate the ad-hoc functionality in the Android operating system. Furthermore, we measure the performance of the ad-hoc connectivity using two different Android phone models. Setting up the real-world experiment setup was supported by the bachelor's thesis of Lars Fritsche [71].

Enabling Wireless Mesh Networks on Android Devices

To enable wireless mesh networks on Android devices, several steps are required. First, the ad-hoc functionality must be reactivated on stock Android devices. As described above, Google Inc. removes this functionality by default from the Linux basis. For reactivation of the ad-hoc functionality, the kernel of the respective device must be recompiled manually with all necessary kernel modules activated.

Second, the Android operating system has a filter integrated to remove ad-hoc networks from WiFi scan results. To bypass this filter, the open source project called Barnacle¹⁸ can be used. Barnacle was originally designed to enable WiFi tethering on Android phones. The core functionality of Barnacle can be used to create WiFi ad-hoc networks.

Third, a multi-hop routing protocol is required to transmit messages through the created ad-hoc network. The open source project OLSRd¹⁹ (short for OLSR daemon) provides a cross-platform implementation of the OLSR routing protocol (cf., Section 3.1.5). It is available for major operating systems, such as Windows, Linux, and Android, and it is used in the Freifunk network in Leipzig, for instance (cf., Section 3.1.3). On Android, OLSRd uses the WiFi ad-hoc functionality to create wireless mesh networks on the network layer (ISO/OSI layer 3) and to route messages between network nodes.

Based on these three steps, we implemented a small framework that constitutes the basis for our peer-to-peer service overlay prototype. The framework provides several methods to setup the wireless network including the SSID and the wireless channel. Furthermore, the framework can be configured to use IPv4 or IPv6 addresses and to automatically generate these addresses from the local MAC address. We also implemented a proof-of-concept app based on that framework to measure network properties, such as, bandwidth and latency. The app combines measurements from the network adapter with readings from the GPS sensor to calculate the distance between communicating network nodes during measurements. The app is also able to block certain devices in the mesh network to emulate multi-hop networks on a small area (otherwise devices would be able to communicate directly with each other).

Measurement Setup

We used our ad-hoc activated devices and the proof-of-concept app to measure the performance of several ad-hoc communication scenarios. We used two different device models:

¹⁸ Homepage of the Barnacle project (visited on May 23, 2014): <http://szym.net/barnacle/>

¹⁹ Homepage of the OLSRd project (visited on May 23, 2014): <http://www.olsr.org>

the Nexus S (released in December 2010) and the Galaxy Nexus (released in November 2011). Both devices are manufactured by Samsung and distributed by Google. Nevertheless, both phones have different hardware, especially, the WiFi chip differs. Hence, we expect ad-hoc performance of both models to be different.

We used two devices of each model and conducted two different experiments to investigate the correlation between the distance and the achievable bandwidth of a direct ad-hoc connection between devices. In the first experiment, we used two devices of the same model as well as of different models and measured the bandwidth for increasing distances. We started with both devices at one spot and then walked away from each other measuring the bandwidth until the wireless link broke. In the second experiment, we used three devices to measure the performance of a multi-hop network. We conducted our experiments on a field outside the city without any interferences from nearby WiFi networks. To measure the distance between devices during the experiment, we used the built-in GPS sensor. We conducted several discrete measurements and calculated average values and standard deviations. For a better comparison, we draw fitting curves based on an exponential function with negative exponent into the plots below.

1-Hop Performance

In our first experiment, we intended to measure the ad-hoc communication performance of homogeneous as well as heterogeneous device constellations. Therefore, we conducted the experiment described above with two Nexus S phones (cf., Figure 53a), two Galaxy Nexus phones (cf., Figure 53b), and finally with the heterogeneous constellation of one Nexus S and one Galaxy Nexus (cf., Figure 53c).

The homogeneous constellation depicted in Figure 53b of two Galaxy Nexus phones can hold the wireless link up over the longest distance. However, the bandwidth of two Nexus S phones proves to be much higher than that of two Galaxy Nexus phones (cf., Figure 53a). Furthermore, the achievable bandwidth between two Nexus S phones does not fall as fast with increasing distance. Already at a distance of 50 meters, the bandwidth of two Galaxy Nexus phones drops below 1 Mbyte/s. Two Nexus S phones achieve a minimum bandwidth of 1 Mbyte/s at a distance of up to 90 meters. The heterogeneous constellation provides even worse performance in terms of both achievable bandwidth and maximum communication distance (cf., Figure 53c). The comparison of the fitting curves from the different constellations presented in Figure 53d shows the performance difference between the investigated constellations. Although the Nexus S is the older device, it achieves a better WiFi ad-hoc performance than the Galaxy Nexus.

Considering packet loss, Figure 53e shows that up to a distance of 50 meters the loss rate increases very fast for all device constellation. This correlates with the fast decrease in bandwidth over the same distance. The packet loss rate for two Nexus S reaches roughly 63% at a distance of 50 meters. For two Galaxy Nexus devices, the packet loss rate reaches roughly 75% at this distance. Again, the old Nexus S phone achieves a better performance than its successor. The worst performance is achieved by the heterogeneous constellation of one Galaxy Nexus and one Nexus S.

Overall, these measurements show the impact of distance on the ad-hoc communication performance of modern mobile phones. Especially within 50 meters, the achievable bandwidth varies heavily. Furthermore, the built-in wireless hardware, antenna placement, and

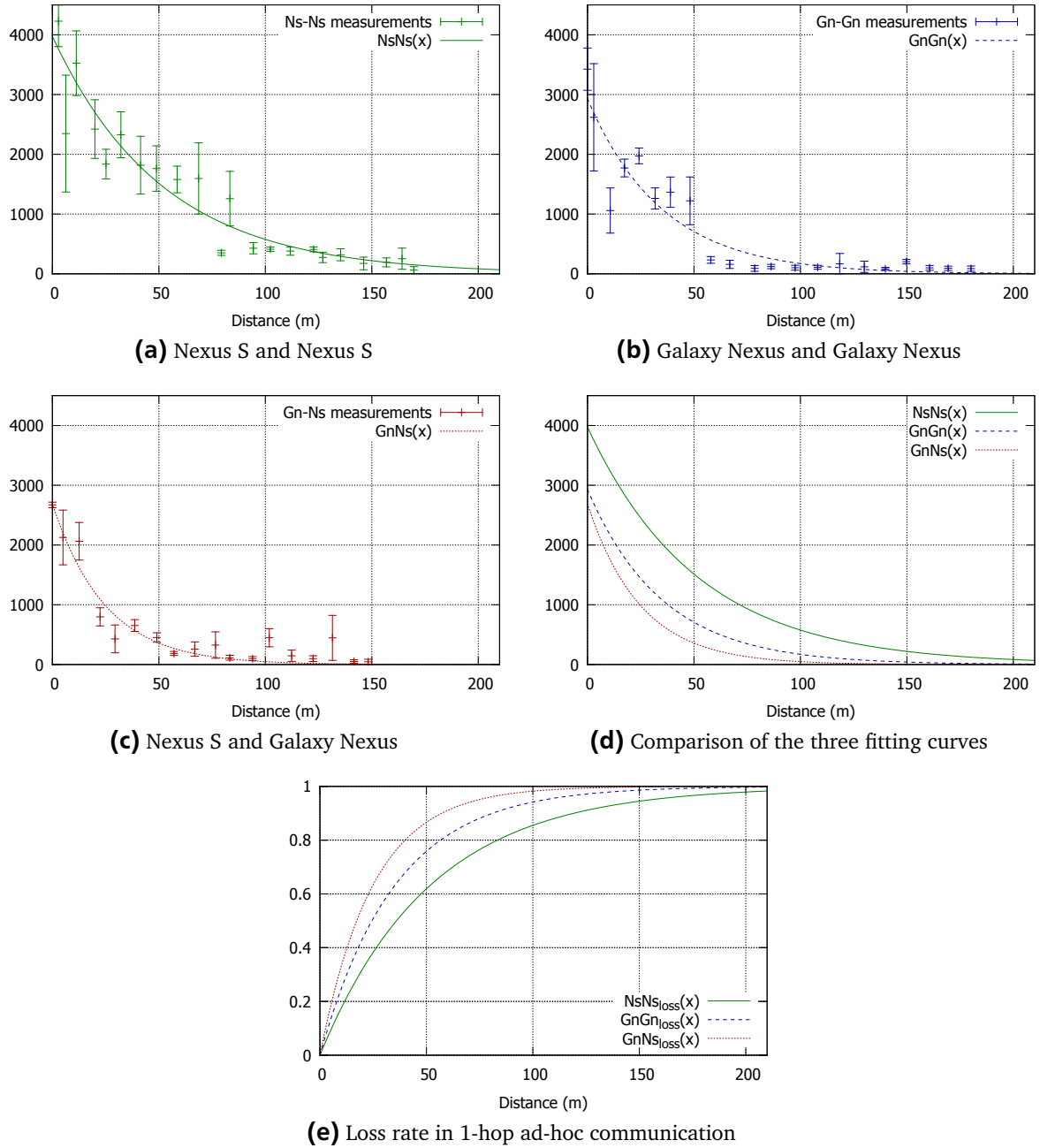


Figure 53: Measurements of 1-hop ad-hoc communication in Kbyte/s

the used materials have an impact on the performance of wireless ad-hoc communication. Bandwidth demanding services should, thus, be consumed over short distances in ad-hoc networks.

2-Hop Performance

In our second experiment, we focused on the real-world multi-hop performance for service provision. For this purpose, we used two constellations of our four available smart phones. First, we used two Galaxy Nexus phones together with one Nexus S phone. Second, we used two Nexus S phones in combination with one Galaxy Nexus phone. The single device was always located in the center between the two devices of the same model. With our proof-of-concept app, we forced the outer devices to always send messages through the center device. This is necessary because for short distances the outer devices were able to communicate directly with each other.

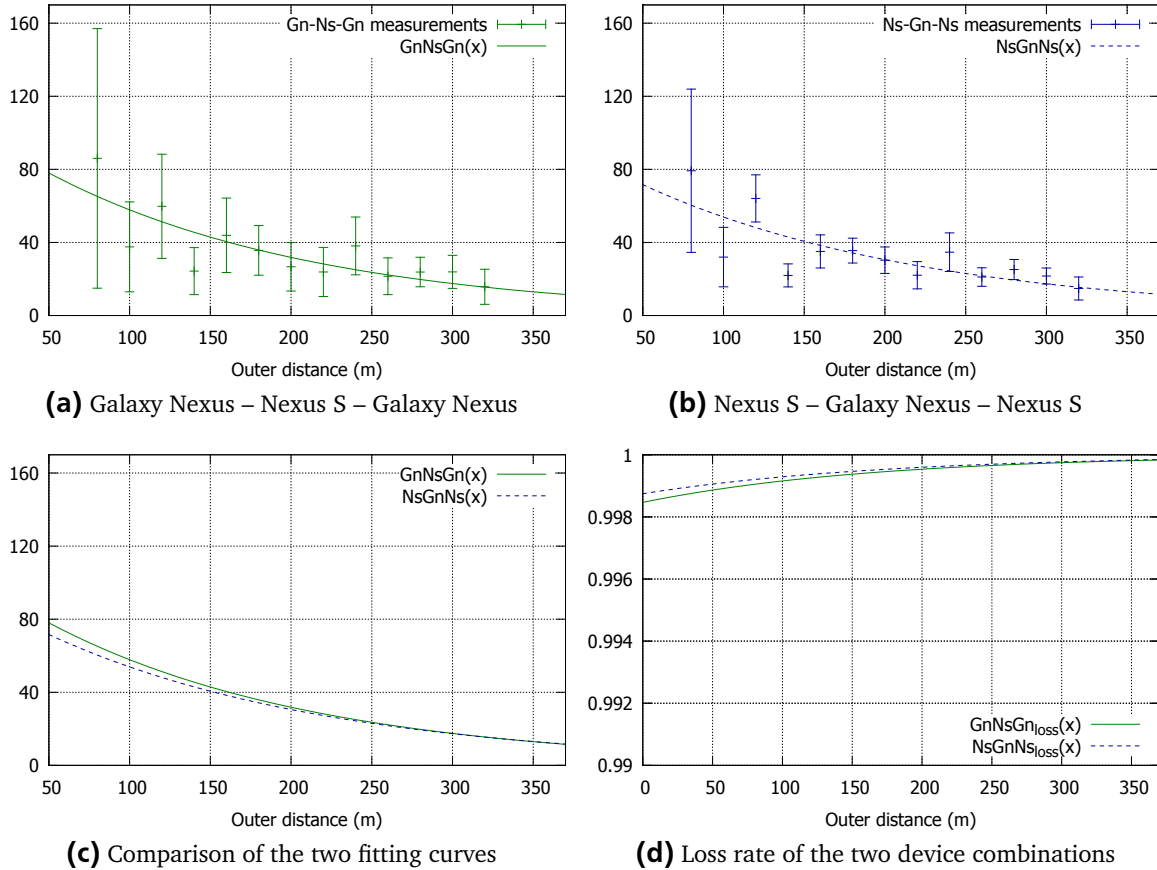


Figure 54: Measurements of 2-hop ad-hoc communication in Kbyte/s

We started our measurements at an outer distance of 80 meters (40 meters for each hop). We chose this distance because lower distances would always result in a direct connection between the outer devices as indicated by the results from the first experiment. Although this might also be true for our setup, we imagine a scenario where the two outer devices

are not in direct line of sight, i.e., they are around the corner of a building with the center device connecting them. Our measurements presented in Figure 54a and in Figure 54b show that with both constellations data rates of roughly 100 Kbyte/s are achievable. As the outer devices move away, the data rate drops down to about 25 Kbyte/s.

The results between the two constellations are not significantly different (cf., Figure 54c). This is easily explained. In both constellations, a Galaxy Nexus sends messages to a Nexus S and vice versa. The scenarios differ only with respect to the ordering of these two events. Therefore, the results are very similar, as expected. The small discrepancy can be attributed to the human inaccuracy during measurements and to the GPS error.

The loss rate, on the other hand, is very high. This can be explained with our previous 1-hop experiments. We showed that, up to a distance of 50 meters between two devices, the loss rate increases drastically. For the heterogeneous constellation, this results in a loss rate of roughly 90%. Since we have two heterogeneous constellations in one sequence, the loss rate combines to around 99%. This is also visible in Figure 54d. This high loss rate is also responsible for the very low bandwidth achieved. Our measurements confirm the findings by Hegde et al. [83] and Niculescu et al. [138]. Both state that the communication performance in mesh networks degrades with the number of hops. Therefore, it is important to keep the amount of data generated by our peer-to-peer service overlay prototype as low as possible.

Assessment

The WiFi ad-hoc performance of today's smart phones seems disillusioning at first glance. The maximum achievable data rate lies around 4 Mbyte/s but only at very short distances. Considering that the wireless standard IEEE 802.11g promises data rates of up to 54 Mbit/s (roughly 6.8 Mbyte/s), this seems rather low. But, our measurements were conducted on the transport layer (ISO/OSI layer 4), thereby, overhead from lower layers were not considered. A net data rate of 4 Mbyte/s is, therefore, reasonable.

Nevertheless, our results indicate that high bandwidths can only be achieved at small distances. Up to a distance of roughly 50 meters, the bandwidth decreases severely (down to 1 Mbyte/s). We were able to achieve a maximum distance of over 150 meters with a homogeneous device setup and with direct 1-hop communication. Bandwidth at such a high distance, however, decreases below 100 Kbyte/s. Furthermore, our measurements for the 2-hop setup showed a severe decrease in bandwidth for larger distances. We, therefore, conclude that bandwidth is a very rare resource in mobile ad-hoc scenarios and in harsh environments in particular. A prototype of our proposed peer-to-peer service overlay must be designed with respect to low bandwidth consumption. This includes data traffic related to service discovery (cf., Section 5.3), overlay management (e.g., joining or leaving the network), and service migration.

5.4.2 Graphical User Interface

As a first feasibility study, we implemented a simple Java-based framework. This framework provides some of the basic features and components described in Section 2.3. We used OpenChord [118], an open source implementation of the Chord DHT by Stoica et al. [185]

as service discovery. Service instances were placed statically in the network. Their locations were stored in the Chord DHT and could be accessed upon service requests. In this first implementation, we did not implement any of the other components. We focused on the service mobility and realized code migration using software agents from the MundoCore framework [7]. A service was then implemented using a MundoCore agent as a starting point.

Afterward, we ported our framework to the Android platform. We call our Android-based framework for mobile service provision *PeerMoS*, short for *Peer-to-Peer Mobile Services*. We built a graphical user interface, enabling users to interact with the system and with provided services. This graphical user interface is presented in Figure 55. The interface has four major parts, which we describe in the following:

Console: The first part of the user interface is the on-device console (not depicted in Figure 55). We use the console to print debug information of the entire framework and also of services. Furthermore, it provides an option to switch between different logging levels (Debug, Info, Warning, Error). Service developers can use this console to debug their services by logging debug information onto the console.

Global View: The second part of the user interface, the global view, contains an overview over all available network information. The global view is depicted in Figure 55a. This view presents information about peers directly connected to the user's device (direct neighborhood) and also about peers that are further away (more than one hop distance). Moreover, the view presents information about remotely running service instances and about service objects available for download. When tapping on a service object, the object is downloaded onto the local device and the user interface switches to the local view. Service objects are small files containing the code of the service (cf., Section 2.4). After downloading them onto the local device, they can be executed within PeerMoS to provide the respective service or to access the user interface stub (cf., Section 2.2).

Local View: The local view, depicted in Figure 55b, presents all service objects available on the local device. Service objects are grouped into “services” and “clients” (e.g., “Voice Chat Service” and “Voice Chat Client”). “Services” are service instances providing the functionality to service consumers through the network. For instance, the “Voice Chat Service” forwards received voice packets to all connected service consumers as discussed earlier. The user interface stub is provided by the “clients”. These service objects are used to access and interact with the corresponding service instance. For example, the “Voice Chat Client” is used to communicate with other service consumers using the “Voice Chat Service”.

Mesh View: The mesh view provides information about the network. It is depicted in Figure 55c. The local device of the user is represented by an icon that shows a person in the center of the view. There are two rings around the user. The first ring represents the direct neighborhood and every peer in one hop distance is placed onto that ring. The second ring represents peers which are in distance of two or more hops.

Service Interface: The service interface appears once the user taps on a service object. Figure 55d presents the service interface for the “Voice Chat Client”. As described above

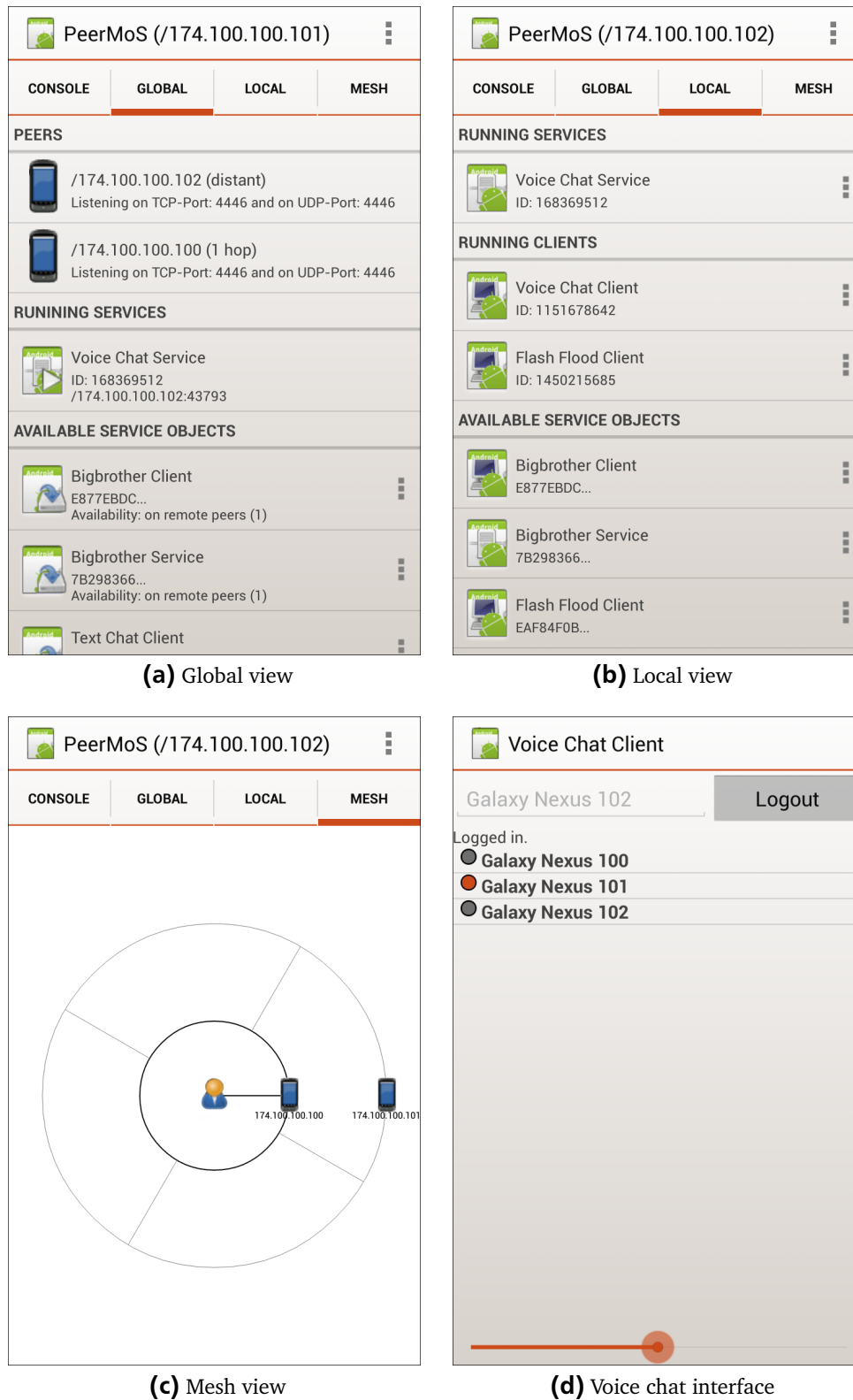


Figure 55: PeerMoS – graphical user interface

and in Section 2.2, this represents the user interface stub to consume the respective service. At the top of the view, the user can enter a pseudonym and log into the corresponding “Voice Chat Service”. Once logged in, the user sees a list of other participants he can talk to. If a user speaks, his icon highlights, signaling that voice packets are transmitted. At the bottom, the user can adjust the sensitivity of his microphone.

5.4.3 Concepts for Smooth Service Migration

As described in Section 3.2.1, service migration causes the service instance to be paused and transferred onto another peer where the service instance is then resumed [206]. This procedure will cause the service to interrupt at least for the time it takes to transfer the service instance and the execution state between peers. During this time, the service functionality is not available to service consumers.

In the previous sections, we further demonstrated that bandwidth is a rare resource in mobile ad-hoc scenarios and especially in harsh environments. Therefore, we argued that the overall bandwidth consumption of PeerMoS should be as low as possible. This includes overlay management, service discovery, and service migration. As we investigated service discovery in Section 5.3, we focus on overlay management and service migration in the present section.

To target both problems, we propose four concepts for continuous service provision through smooth service migration. These concepts include soft handover, persistent socket connections, efficient serialization, and an efficient network protocol. The last two concepts especially target the reduction of bandwidth consumption.

Soft Handover and Persistent Socket Connections

Soft handover and persistent socket connections constitute the basis of our smooth service migration concept. Both concepts are required to eliminate service interruptions during service migration. Although the concepts are targeted at migration of any service type, we use the voice chat service again as an example service. Migration of a voice chat service is only feasible if the communication does neither stutter nor disconnect. The combined concept of smooth service migration is depicted in Figure 56. The figure depicts two service consumers (Q_1 and Q_2) that send (voice) data to a service instance (P_1). The service instance then sends the (voice) data to all other service consumers but the source.

We do not pause the service instance during migration. Instead, we replicate the service instance onto another peer and synchronize the execution state afterward. This means that the service code and the execution state are copied from peer P_1 to peer P_2 and executed remotely. During the entire migration procedure, state changes are propagated to the new service host using synchronize messages. Once the service instances on both peers are synchronized, the service instance on the new peer (P_2) takes over the service provision. It sends a take over message to the old service instance (P_1) and to all service consumers. The old service instance then disconnects remaining users and exits. At this point, all service consumers send data only to the new service instance on P_2 . This basically implements a soft handover protocol.

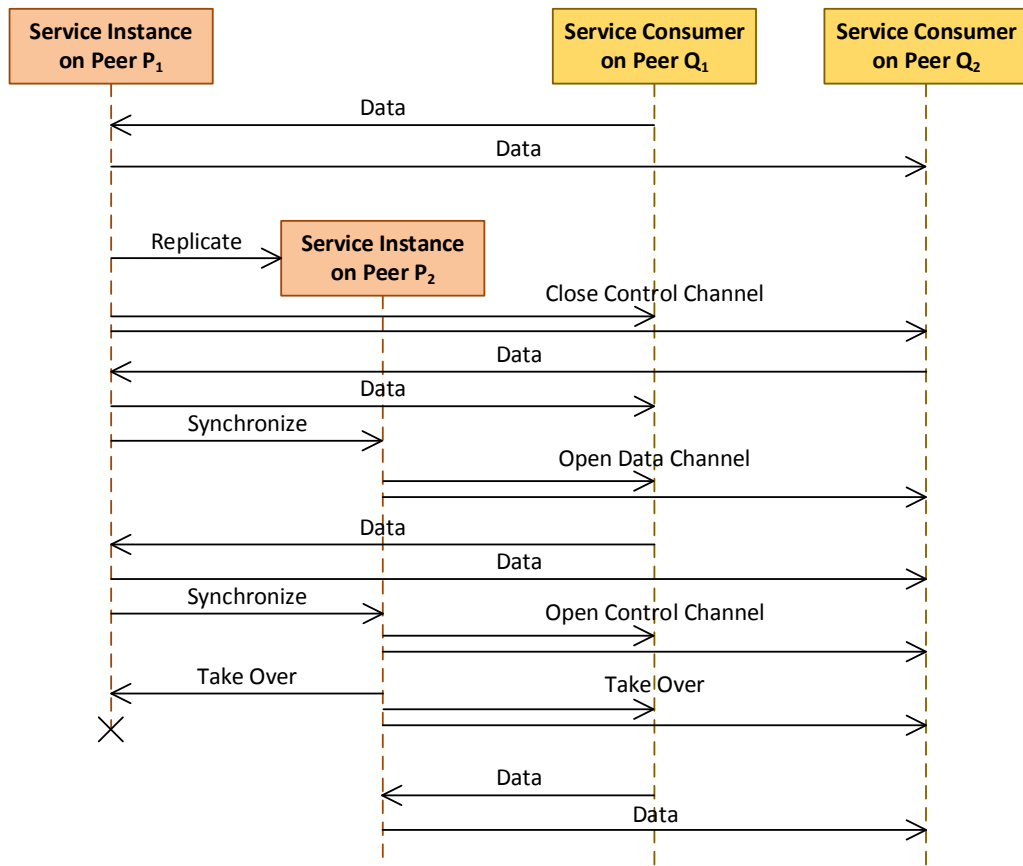


Figure 56: Sequence diagram of our smooth service migration concept

Furthermore, not only the execution state needs to be transferred and synchronized but also the connections between service consumers and the service provider. To realize this, we developed *persistent socket connections* inspired by the MobileSocket approach by Okoshi et al. [144]. We use two communication channels: a TCP channel for control messages and a UDP channel for data transmission. During the migration process, the control message channel is paused, but the data channel remains open. The old service instance (on P₁) initiates this by closing the control channel for all service consumers. The new service instance (on P₂) is able to connect to all service consumers because their addresses are stored in the synchronized execution state. After the new service provider P₂ sends a take-over message to all reconnected service consumers, they start to transmit their (voice) packets to P₂ and disconnect from P₁. This introduces timing problems because there might be service consumers still sending (voice) packets to the old service provider P₁, while other service consumers already send their packets to P₂. These packets are then propagated from P₁ to P₂ by sending synchronize messages. Furthermore, we introduce sequence numbers to allow for a seamless transmission of these delayed packets. Experiments conducted with our prototype were free from voice interruptions during the migration process.

To migrate service instances during runtime, the so-called serialization is used. Serialization is the transformation of a program or an object into a flat, persistent, and usually binary representation. Deserialization is the transformation of that representation back into a program or an object. This is necessary to transfer the program or the object to another computer in a bit-stream. Java already provides a serializer to realize service migration. However, the Java serialization has bad performance in both speed and resulting packet size. Therefore, we investigate means to optimize service migration through efficient serialization. Research on efficient serialization was supported by the bachelor's thesis of Felix Hammacher [81].

In the first versions of PeerMoS, we used Java to serialize service instances and also all network messages. This resulted in around 800 bytes data to be transmitted for a simple HEARTBEAT message – a message without any content. This is because the Java serializer includes the class name string of the corresponding message and more information into the data packet. All that information is needed to correctly deserialize the data packet after receiving it on the other node. Even compressing the serialized data with the *gzip* algorithm still resulted in around 400 bytes. We, therefore, compared different serialization frameworks with each other. We identified the *Kryo*²⁰ serialization framework to be the most suitable candidate for our purpose. The project goals of Kryo are speed, efficiency, and an easy to use API.

Although performance of Kryo was good during our initial tests, the framework was lacking some required features. The biggest problem was that Kryo was not able to deserialize classes that lack a zero-argument constructor. When deserializing a class, it is unknown to Kryo what constructor to use to instantiate the respective class. Therefore, the framework requires a constructor that has no arguments for class instantiation. All fields are then set after instantiating the respective class. The Java serializer can handle such classes because it uses internal methods to instantiate the respective class and to set all necessary fields. In a standard Java virtual machine, Kryo can also handle classes lacking a zero-argument constructor by using a third party library. On Android, however, this library was not available at the time we optimized Kryo for PeerMoS.

We implemented a custom method to handle classes lacking a zero-argument constructor. If this method recognizes such classes, native methods are used to instantiate them. This is analog to the Java serializer. During serialization, the respective classes are recognized and the appropriate instantiation strategy is stored in the serialization database. This database is then used during deserialization to correctly instantiate serialized classes.

Finally, we used four concepts in our improvement of the Kryo framework to further increase the performance of the serialization and deserialization process:

Reuse existing objects: Because the instantiation of new objects needs much time, we reuse old objects that are not needed anymore. We clear the fields of such objects by calling a reset method on them.

Initialization: During serialization, Kryo stores information about serialized classes in a database. By initializing the database at program start with all available classes, we

²⁰ Homepage of the Kryo project repository (visited on May 17, 2014):
<https://github.com/EsotericSoftware/kryo>

can reduce the serialization time for already registered classes. Unknown classes have to be registered during the serialization and deserialization processes.

Use efficient data structure: To manage the serialization database, an efficient data structure is required that allows fast access on individual items.

Reuse information: Since the creation of new information takes time, we try to reuse all information available. For instance, when adding information about a class to the Kryo database, we only adjust the corresponding index if the class already exists in the database.

Assessment of the Enhanced Serialization Framework

To evaluate the performance of our extended version of the Kryo serialization framework, we extended PeerMoS to measure the processing times for serialization and deserialization and the resulting packet sizes after serialization. We used two Galaxy Nexus smart phones for our measurements and the “Voice Chat Service” as the service instance to be migrated. We conducted two experiments. In the first experiment, the voice chat service instance is started and then transferred to the second device. In the second experiment, we also connect two service consumers to the service instance. We executed both experiments ten times and measured the processing times of the serialization and deserialization processes as well as the resulting packet sizes. The results of our measurements are presented in Table 13. For the serialization and deserialization processes, we calculated the average times and the standard deviations.

Clients	Serializer	Serialization	Deserialization	Packet size
0	Java	35 ± 15 ms	32 ± 3 ms	1997 bytes
	Java + gzip	46 ± 11 ms	41 ± 5 ms	913 bytes
	Kryo-Generic	11 ± 2 ms	14 ± 2 ms	126 bytes
2	Java	67 ± 16 ms	54 ± 9 ms	3686 bytes
	Java + gzip	72 ± 21 ms	74 ± 14 ms	1550 bytes
	Kryo-Generic	22 ± 7 ms	23 ± 4 ms	727 bytes

Table 13: Serialization measurements of the voice chat service

Our results show that the Java serializer generates the largest packets and requires the longest period of time for serialization and deserialization. Although the gzip algorithm is able to further compress the resulted packets, this approach cannot compete with our extended Kryo serialization framework (called *Kryo-Generic*). In terms of speed, Kryo-Generic is able to reduce the time for serialization and deserialization by roughly 66% compared to the Java serializer. As for the packet size, it heavily depends on the data that is serialized. Without service consumers, Kryo-Generic generates packets with only 6% of the packet size compared to the Java serializer (reduction of roughly 94%). However, with two connected service consumers, the resulting packet size can “only” be reduced by roughly 80% compared to the Java serializer.

In conclusion, we can say that if packet size and processing times are of importance for the performance of serialization, the Java serializer should be omitted in general. Other serialization frameworks can provide much better performance. For our PeerMoS prototype, these aspects are very important. The Kryo-Generic serialization framework is capable of efficient serialization and is, therefore, used to migrate service instances in PeerMoS.

Efficient Network Protocol

As introduced in the previous section, we used Java to serialize all network messages in the first versions of PeerMoS. This resulted in heavy bandwidth utilization in terms of management-related data traffic. To reduce this data traffic, we created our own binary protocol to transmit PeerMoS messages. With our protocol, PeerMoS messages have a minimum size of 15 bytes for an IPv4 network and a minimum size of 39 bytes for an IPv6 network. Figure 57 depicts a schematic overview of our binary network protocol, including the seven PeerMoS message types.

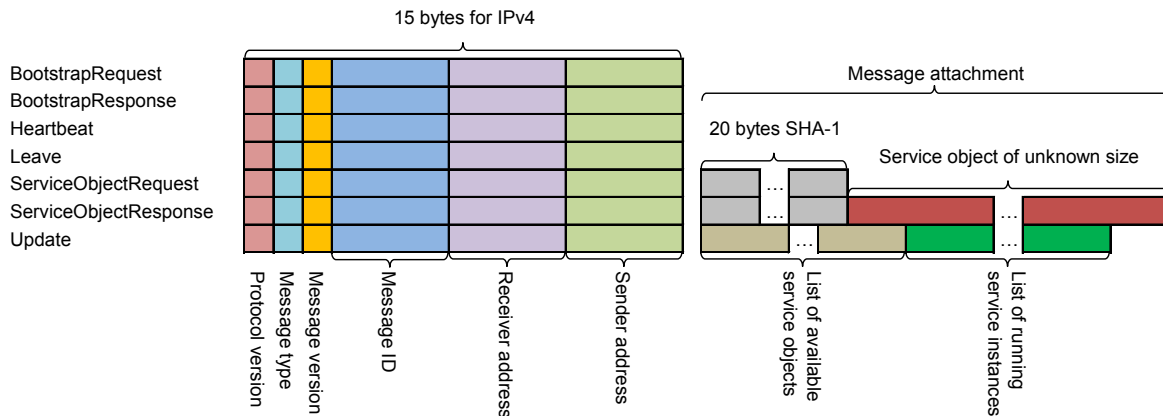


Figure 57: Binary network protocol of PeerMoS in an IPv4 network

Bootstrap messages are exchanged when new participants join the network. Joining participants send a `BOOTSTRAPREQUEST` message to a member of the network who then acknowledges the request with a `BOOTSTRAPRESPONSE` message. If a user exits the PeerMoS app, a `LEAVE` message is sent to all neighbors to inform them about the app termination and the user's disappearance. `HEARTBEAT` messages are sent periodically to inform all neighbors that the peer is still alive and available. If a user wants to consume a service and requests a service object, a `SERVICEOBJECTREQUEST` is sent to an appropriate peer. This peer then sends a `SERVICEOBJECTRESPONSE` message back with the service object file attached to the message. Both messages use a 20 bytes SHA-1 hash to identify the respective service object. Every time a change happens to the list of known service objects or running service instances, an `UPDATE` message is sent to all neighbors to refresh the list of available services. This message contains two lists with all available service objects and all running service instances, respectively.

Finally, the migration of service instances is also done using a PeerMoS message. This message also uses the same 15 bytes header as the other messages and is called `INSTANCE` message. After serialization, the serialized service instance is attached to the `INSTANCE` message.

and transferred to another peer. Since the message type does not depend on the serializer, different serialization frameworks can be used.

5.5 Summary

In the present chapter, we described our proposed peer-to-peer service overlay concept. We started off in Section 5.1 with the description of our system model. We discussed how all components work together to achieve service provision in a peer-to-peer network. In particular, we focused on the design and implementation of the service placement and the service discovery components. Furthermore, we presented additional components relevant to our simulation-based investigations of service placement and service discovery.

In Section 5.2, we conceived four different service placement approaches. Two approaches use the quality metric *hop count* and two approaches use the quality metric *latency* to relocate a service instance. The goal of these approaches is to provide a consistent service quality to all connected service consumers. We investigated all four placement approaches in a disaster recovery setting using a voice communication service. Our simulation-based evaluation confirmed the good overall performance of the state of the art tree placement approach by Oikonomou et al. [141]. Especially in the single group setup, this placement approach provides good performance in terms of low message loss rates, few service migrations, low mouth-to-ear delay, and low hop counts. However, the tree placement did not perform as well in the command center setup typically found in disaster recovery scenarios. We demonstrated that the tree placement approach achieves almost the same performance as a static placement on one of the first responders in the field. Considering the command center, almost 50% of the voice packets from the first responders never reach the command center in our simulation setup.

Furthermore, our conceived *hop count placement* approach was able to achieve consistent service quality for all group members as well as for the command center by minimizing the hop count for all connected service consumers. In the command center setup, the hop count placement achieved good performance in terms of consistent hop counts and mouth-to-ear delays for the command center and all group members. It achieved this goal with only 20 to 30 service migrations per hour and moderate overall message loss of up to 20%. In the single group setup, our hop count placement approach achieved comparable results as the state of the art tree placement approach. As for the latency-based placement approaches, we came to the conclusion that latency is not a suitable quality metric to base service placement decisions on.

In Section 5.3, we presented our evaluation of different service discovery approaches. We especially focused on the interdependency of the service placement and the service discovery components as service placement introduces additional dynamics through service mobility. Our simulation-based evaluation of six different service discovery approaches revealed surprising results. Although we selected three state of the art service discovery approaches designed for mobile ad-hoc network environments, we found their performance to be disappointing. All three approaches were able to handle small networks with only few services. However, with increasing number of peers and services, these approaches heavily increased discovery-related data traffic and were not able to successfully discover mobile services.

Most surprisingly, the naive request flooding approach performed best in terms of discovery performance and data traffic efficiency. Neither peer movement nor frequent service

migrations were able to have a drastic negative impact on the discovery performance of this simple approach. Data traffic generated by this discovery approach was smallest under all investigated conditions. Discovery performance was very high under almost all investigated conditions. Therefore, we recommend the request flooding approach to discover services in peer-to-peer service overlays for harsh environments.

Finally, we presented our concepts for smooth service migration in Section 5.4. We presented our real-world experiment setup to investigate smooth service migration and demonstrated that bandwidth is a rare resource in wireless multi-hop networks. Hence, service migration must not only be smooth but also efficient in terms of bandwidth consumption. We further presented the graphical user interface of our service overlay prototype, called *PeerMoS*. Within *PeerMoS*, we realized our concepts for smooth service migration: soft handover, persistent socket connections, efficient serialization, and an efficient network protocol. The first two concepts aim at eliminating the interruptions and service unavailability during service migration. The last two concepts aim at reducing data traffic in our peer-to-peer service overlay prototype.

6 Simulation Models and Tools for Harsh Environments

To provide versatile software services in harsh environments (e.g., disasters), a resilient communication infrastructure as well as a peer-to-peer-based communication system are required. We introduced a novel approach to enhance the communication infrastructure in disaster-affected urban areas by using public and private WiFi routers in cities to create an emergency mesh network, the CityMesh (cf., Chapter 4). Furthermore, we introduced our system model for service provision in harsh environments, the peer-to-peer service overlay (cf., Chapter 5). We investigated critical system components of that system and proposed a novel service placement approach to ensure consistent and high service quality for all connected service consumers. In the present chapter, we introduce our simulation framework for disaster recovery scenarios covering all four system layers discussed in Chapter 2: the geographical region, the communication infrastructure, the participants, and the communication system (the peer-to-peer service overlay). The contributions in the present chapter can be summarized as follows:

1. We propose our simulation models to investigate communication systems in harsh environments. Our study conducted with experts from rescue departments revealed that movement of rescue workers depends on various factors: the terrain and the current disaster situation, the mission strategy of the respective rescue workers, and the orders received from the command centers. Hence, a detailed world model as well as an integrated mobility and communication model are required to evaluate the effectiveness of communication systems in realistic harsh environments.
2. We present our integrated simulation framework *DisVis* with support for detailed agent-based mobility and communication models. To enrich the simulation with realistic network models, we define an interface to connect our framework with state of the art network simulators.

The contributions in the present chapter were published in the proceedings of the International Conference on Mobile Services, Resources, and Users in 2013 [155]. Our publication won the best paper award.

The remainder of this chapter is organized as follows: In Section 6.1, we present our simulation models for disaster recovery simulations. This includes the detailed world and the mobility and communication model. These models cover the movement, behavior, and communication of first responders during simulated rescue missions. In Section 6.2, we present adjustments of our models based on information we extracted from expert knowledge. Afterward, we introduce our integrated simulation framework *DisVis* in Section 6.3. This also includes the interlink between the mobility simulator and the network simulator. Finally, we summarize this chapter in Section 6.4.

6.1 Simulation Models

In Section 3.3, we discussed simulation models for mobile communications research as well as for disaster recovery. In many studies, movement of participants is based on a random mobility model. In reality, however, participants (i.e., rescue workers) move according to certain properties in their surrounding environment and according to their specific behavior. Especially in harsh environments, we argue that the geographical region greatly influences the movement of all participants. In disaster recovery scenarios, for instance, it is important for paramedics to know the locations of injured patients in the field. Based on this information, they can move to the patients for treatment. Furthermore, rescue workers use vehicles to move on streets to, from, and between incident sites. Therefore, information about the world is essential for disaster simulation and for realistic movement patterns in general.

In conventional simulation setups, the communication between participants is not considered to have an influence on their movement. We believe this loop should be completed and propose the consideration of the communication between participants to influence their movement, as well. Therefore, the mobility model must not only generate output (movement traces) for the communication simulation but, at the same time, it must also accept input from the communication simulation. This loop suggests that the mobility model should consist of more than movement aspects only. Therefore, we propose a mobility model for harsh environments that consists of two components to model all important aspects of the movement of participants and their communication. Besides moving, participants also need to interact with each other and their surrounding environment. Therefore, we design our mobility model to contain two components that integrate into our mobility model for harsh environments. These components are the *movement* and the *communication and interaction* components. Figure 58 presents an overview of the required simulation models for communications research in harsh environments.

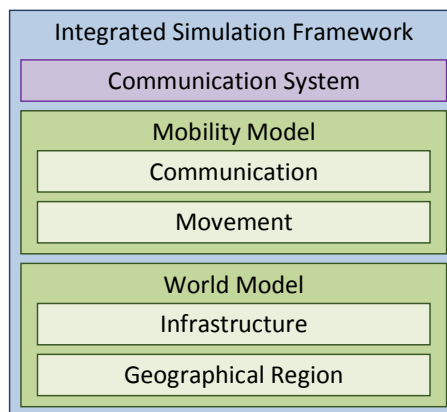


Figure 58: Required simulation models for communications research in harsh environments

Below, we present our simulation models in more detail. First, we describe our world model in Section 6.1.1. Afterward, we present the movement component of our mobility model in Section 6.1.2 followed by the communication and interaction component in Section 6.1.3. Research on mobility and communication models was supported by the bachelor's thesis of Pascal Weisenburger [197].

6.1.1 World Model

The world model constitutes the basis for mobility simulation. The world model consists of the geographical region where participants move in and of the communication infrastructure participants use to transmit messages among each other. Typically, this world is modeled as a two-dimensional plane and participants are placed on this plane (cf., Section 3.3.1). In some cases, also obstacles are placed on this plane to model the world more realistically. During the mobility simulation, the participants are then moved on this plane, thus, generating movement traces.

We use OpenStreetMap²¹ [80] data to generate a realistic world model for our simulations. This data provides rich information about streets, buildings, fields, woods and other geographical properties. If vehicles move on streets, the speed limits of the respective streets can be used by the mobility model to adapt the speed of moving vehicles. Buildings from this data set can be considered for simulating fires in apartments. Furthermore, the data set also provides information about the headquarter locations of the individual rescue departments. This includes fire stations, police stations, and hospitals. This information can be used in the simulation as targets for ambulances to transport injured to, for instance. All this data can be used by the mobility model to generate realistic movement traces for disaster recovery simulation.

Based on the geographical data, seats of fire can be placed. In our world model, we assume the fire to spread at a rate of $0.25m^2$ per minute (according to DIN 18232). In parallel, fire fighters can extinguish the fire. Furthermore, we use the state of the art work by Aschenbruck et al. [16] (cf., Section 3.3) to create the three virtual disaster zones surrounding the fire places: the incident site, the treatment zone, and the transport zone. We use the information about hospitals provided by the OpenStreetMap data to create the hospital zones. These zones are then used in the rule sets of our software-agent-based mobility model. Later in the present chapter, we present a graphical representation of the world model (cf., Figure 63 in Section 6.3.1). In the following section, we present the movement and communication components of our mobility model.

6.1.2 Mobility Model: Movement

The movement component of our mobility model is based on the concept of software agents [24] and also constitutes the basis for our mobility model. This means, every participant (i.e., first responder) is represented by a software component that generates the movement patterns for that particular node. Agents, on the other hand, are instances of an agent type. Every agent type contains a set of rules that describe how to react in specific situations and on different events. In the case of disaster recovery, agent types can be *fire fighter* and *police car*, for example. The individual fire fighters or police cars are then instances of the *fire fighter* or *police car* agent types, respectively. Figure 59 depicts the different agent types in our model.

Each agent has limited knowledge about the simulated world including other agents and the world model. In particular, each agent knows about the geographical map, that is the road network and the locations of the associated institutions such as hospitals or police

²¹ Website of the OpenStreetMap project (visited on May 19, 2014): <http://www.openstreetmap.org>

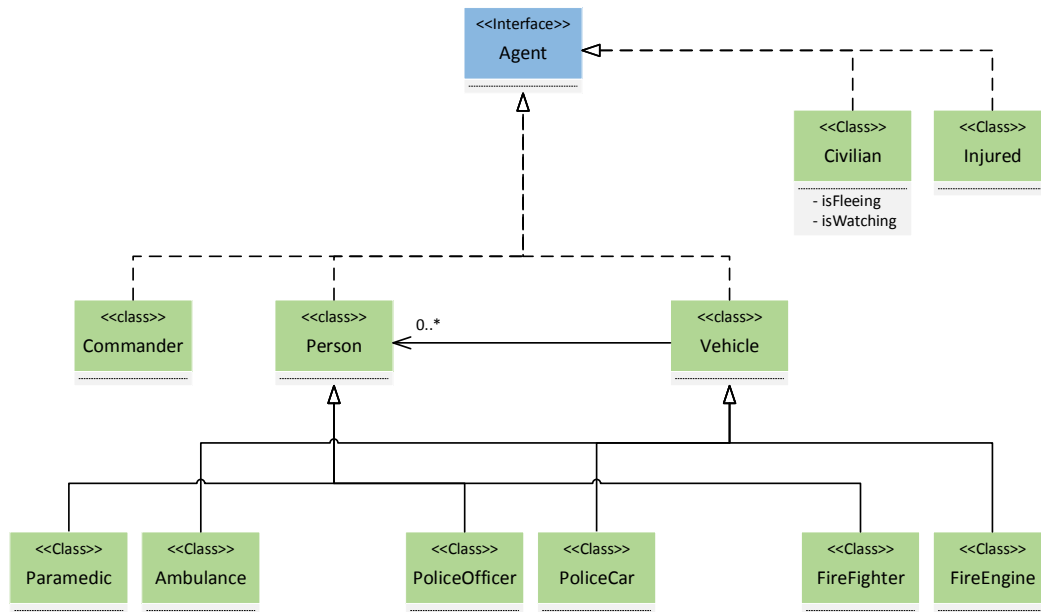


Figure 59: Agent types of our mobility models for disaster recovery

and fire stations. At the beginning of the simulation, every first responder agent has initial information about the type and location of the disaster areas, injured people in these areas, the partitioning into disaster zones, and the locations of their colleagues. This basically represents a state directly after the mission briefing.

During the simulation, every agent chooses specific targets in the world based on its currently available information. The agent then moves towards that target considering the geographical information (such as streets) or it retains the current position for a specified time span. The target choice determines the movement pattern characteristic to a particular agent type. Our behavior-based mobility model differentiates not only between the various types of first responders, such as *police men* or *fire fighters*, but also between the ways of moving, such as *by motor vehicle* or *by foot*. Furthermore, vehicles are able to transport persons. Therefore, agents are moving specifically to their designated roles in order to generate behavior-based movement patterns as realistic as possible.

Task Assignment

The movement of the agents roughly follows the simple high-level actions *repelling*, *attracting*, *oscillating*, and *immobile* from the generic event- and role-based mobility model described by Nelson et al. [136] (cf., Section 3.3.1). However, the target selection of an agent depends on a substantially more fine-grained behavior model. In our model, tasks specifically assigned to the respective agent type are simulated at the level of the agents:

Fire fighters: Fire fighters have two primary tasks. First, casualties have to be transported from the *incident site* to the *treatment zone* where the patients can be treated by paramedics. This is because only fire fighters have the special equipment necessary

to operate inside the incident site. The second task is to fight the disaster, i.e., to extinguish fire.

Paramedics: The task of medical personnel during rescue missions is to bring injured people out of danger to the *treatment zone* and to treat and stabilize them there. Afterward, they bring the patients to the *transport zone*, where they are picked up by ambulances to be transported to a *hospital*. The paramedic then heads for another patient in order to stabilize him and to prepare him for transportation.

Police: The police has the task to secure the disaster-affected areas. This means that the traffic hubs (cross roads) need to be secured to prevent civilians from entering the disaster area. Police officers have to patrol between several intersections and expel civilians who are already within the disaster area.

Civilians: The movement of civilians is based on the gravitational mobility model described by Nelson et al. [136] (cf. Section 3.3.1). This allows civilians to respond to the presence of multiple disaster events. Agents can flee from several independent disaster events or approach them. This depends on the agent type of particular civilian agents. We implemented watching, fleeing and injured agent types. Watchers try to watch the recovery workers during their work and the fleeing agents flee from the disaster area. Injured civilians do not move alone but are always transported by either a fire fighter, a paramedic, or an ambulance.

Figure 60 shows an example rule set for the paramedic agent type. Rule sets for other agent types are alike and contain the behavior specific to the respective agent type. In general, this agent-based concept provides the opportunity to simulate other scenarios as well, for instance, a public transportation system. Buses, trams, and trains in cities can be modeled using our agent-based mobility model. This can be done by implementing the rule sets for such transportation agent types.

6.1.3 Mobility Model: Communication and Interaction

The rules above show that the movement of agents and their behavior are directly based on interaction with the surrounding world and also with other agents. Fire fighters extinguish fires in the environment and paramedics carry injured people to ambulances. Ambulances then transport injured people to the hospitals. All agents use the underlying street network from the world model to move in the geographical region.

But the interaction between agents contains more than just transporting other agents through the world. Moreover, first responders can send requests to other rescue workers if they need help or a transportation vehicle. For example, paramedics request ambulances to allow patients to be transported to a hospital. An ambulance that is available for transportation can then acknowledge the request and move to the requested location. The transportation of first responders to the disaster area and back to the headquarters is carried out in the same manner. Also, first responders inform their colleagues about their new position as soon as they arrive at a new location to complete an assigned task, e.g., treat a patient. This helps to keep the information about the positions of colleagues up to date for all first responders.

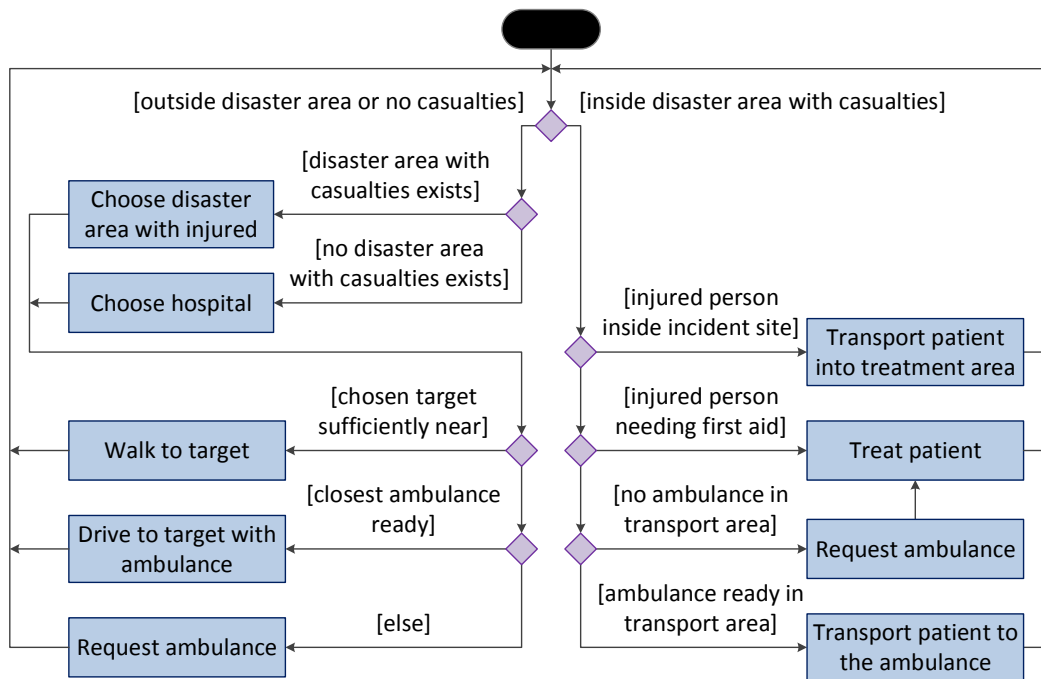


Figure 60: Rule set of paramedics

Finally, first responders inform their colleagues and the commander (i.e., the command center) about the progress of the rescue mission. Especially after a fire was extinguished, the fire fighter agents inform their colleagues about the finished task. Also, paramedics inform their colleagues about rescued patients. All message types are summarized in Table 14. Based on this information, first responders know what tasks are remaining and also who is available for a new task assignment. The agents use all this information to decide who should be assigned to what disaster event next.

Message type	Description
Position Report	Inform colleagues about the own current position when arriving at a target.
Target Cleared	Inform colleagues and the commander whenever a fire was extinguished or a patient was rescued. This message further requests a new target.
Assistance Needed	Respond to a received Target Cleared message. Includes all remaining fire places to be extinguished or patients to be rescued.
Transport Needed	Request a vehicle for transportation.
Transport Confirmed	Respond to a transportation request if available.

Table 14: Message types of first responder agents

Furthermore, all communication of rescue workers is carried out hierarchically. A vehicle and a small number of rescue workers form a task force. The communication is always carried out inside this group. All groups are then organized by a commander of the corresponding department. All commanders are connected together to exchange information

and to pass requests and orders on to groups and to individual rescue workers. Figure 61 depicts an example information flow from a paramedic to all other first responders in the field.

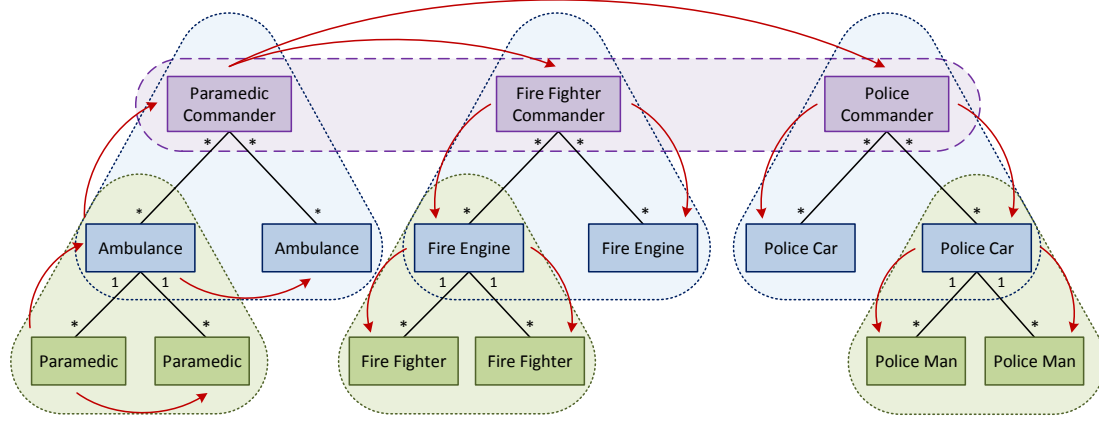


Figure 61: An example message flow in the communication hierarchy of first responders

This shows how the movement of agents depends on the communication between them. Conversely, the movement of the agents also has an influence on their communication. If a real network based on wireless communication is considered for message transportation, the agents need to be in transmission range of each other for messages to arrive at their destinations. So the network simulation cannot be launched after the mobility simulation, but both simulations have to be executed together in parallel to take mutual influences into account.

6.2 Fine Adjustment of Simulation Models

To fine adjust our simulation models we were drawing on expert knowledge. We created a questionnaire targeted at experts from police departments, fire fighting departments, and medical facilities. With this survey, we intended to extract different types of information. This resulted in three similar questionnaires, one for each of the first responder groups listed above. Each questionnaire was organized in three parts to get information about:

- i) activities of first responders during missions,
- ii) the communication between them, and
- iii) the four disaster zones (cf., Section 3.3).

Our call to fill out the survey was followed by 84 individuals most of which already participated in disaster recovery missions. This number divides into 44 individuals from medical facilities, 32 individuals from fire fighting departments, and 8 individuals from police departments. The low number of participating members from police departments indicates that their role during disaster recovery missions might not be as important as that of the other groups. This was also reflected in the answers given by both the participants associated with the police departments and the participants associated with the fire fighting

departments. Usually, fire fighters are the first to arrive at the incident site. Hence, they take over certain roles associated with the police. The main activities associated with the police are the investigation and the criminal prosecution. Both are usually long term activities and start after the disaster recovery mission is finished. Below, we discuss the findings of our study and how we further improved our simulation models.

Activities

In the first part of our survey, we targeted our questions at the atomic activities implemented in the rule sets of our mobility model (cf., Figure 60). These activities model the abstract behavior of the three first responder agent types. One goal was to verify our assumptions about the priorities and the rules for these atomic activities. The other goal was to identify important but missing activities. Our assumption about the priorities of atomic activities was accurate in general. However, the activity of *examining the situation on-site* is not modeled explicitly in our rule set. But, as stated before (cf., Section 6.1.2), the agents know about the situation in the disaster areas at the beginning of the simulation. This knowledge is incorporated into their decisions and activities. Hence, this activity is modeled implicitly in our mobility model.

With our questions about the activities, we also intended to get details about these activities, for instance, the typical walking and driving distances. We also tried to get a rough estimate about the average times paramedics need to stabilize patients and fire fighters need to extinguish a fire in apartments. In general, these details can only be estimated very roughly and depend on various factors. However, we found that typically distances of up to 400 meters are walked by foot. To stabilize patients, paramedics assess 2 to 10 minutes. Fire fighters assess roughly 30 to 90 minutes to extinguish a fire in apartments, although, the participants explicitly pointed out that the time is highly situation-dependent and cannot be generalized. Nevertheless, we assume fire fighters in our model to extinguish fire at a rate of $1.5m^2$ burning area per minute.

Communication

In the second part of our study, we checked our assumptions made about the relevance of communication between first responders. This also included the organizational structures. We found that our assumptions about the highly hierarchical organizational structures of first responders is accurate in general and that communication is very important in disaster recovery missions. In fact, most activities are only executed by orders, which are passed along the hierarchical structure.

Furthermore, our survey exhibited the explicit importance of small first responder groups. For example, a team of fire fighters and a fire engine form a group that moves and operates together. The same applies to police officers and paramedics. This affects the movement of every unit that is part of a group. It also impacts the communication, which is simulated hierarchically so that the members of these groups primarily communicate among themselves and only the group leaders communicate between different groups. We implemented this behavior into the rule sets of our simulation model.

Finally, we verified the assumptions made about the partitioning into different disaster zones and their sizes as described by Aschenbruck et al. [16] (cf., Section 3.3.1). The existence of different zones was confirmed by the participants, as expected. Usually, first responders define locations and sizes of these zones before the recovery mission is started. This is also captured in our simulation model. The size of the incident site highly depends on the affected region (e.g., 20 to 50 meters perimeter for a burning apartment or 100 to 150 meters perimeter if a factory is burning). In our simulation model, we use 20 meters around the affected region, for instance, a building with a burning apartment.

Usually, the treatment zone is rather small and can be combined with the transport zone into one zone under certain circumstances. Depending on the situation, the treatment zone is at most half in size compared to the incident site. The transport zone, on the other hand, is rather large and can be up to twice as large as the incident site. Usually, the transport zone relies on qualified locations for transport vehicles to access the zone. These locations should be as close as possible to the treatment zone. In our simulation model, we use at least 20 meters for the treatment zone and at least 50 meters for the transport zone. All three zones are adjacent: the treatment zone shares a common border with the incident site on the one side and with the transport zone on the other.

In addition, the initial partitioning of the disaster area into zones can change over time when disasters are spreading or are averted. This fact is also reflected in our simulation model. Zones change dynamically with the progress of the simulated recovery mission. These details about the zones were incorporated into our simulation model after the survey.

6.3 Integrated Simulation Framework

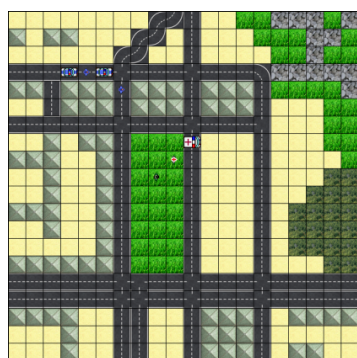
In the previous sections, we proposed simulation models to investigate new communication systems in harsh environments. We presented the world model that constitutes the basis for realistic simulations of harsh environments. It also includes the communication infrastructure. Furthermore, we presented our mobility model for rescue workers in disaster recovery scenarios. We argued that for realistic simulations the movement of participants and the communication between them must influence each other since most activities are only executed by command. Hence, the movement and communication components of our mobility model must be able to interact with each other during the simulation. To realize the interaction between the components, an integrated simulation framework is necessary. In the present section, we introduce our approach for an integrated simulation using disaster recovery missions as an example.

Our realization of an integrated simulation consists of two parts. First, we introduce our integrated framework for disaster simulation, called *DisVis* short for *Disaster Visualization* (Section 6.3.1). This framework includes all of our simulation models described above: the world model that constitutes the basis for the entire simulation and the mobility model that generates the movement of participants (i.e., rescue workers) as well as the communication between them. Second, we define an interface to link our simulation framework with a network simulator to realistically simulate the network communication (Section 6.3.2). As a proof of concept, we implemented this interlink using the network simulator Peerfact-

Sim.KOM [183]. Research on the integrated simulation framework was supported by the bachelor's thesis of Denis Caruso [41].

6.3.1 Disaster Simulation Framework: DisVis

In Section 3.3.3, we identified the first response communication sandbox by Bradler et al. [36] to be a good starting point for our disaster simulation framework. The sandbox provides a graphical user interface to create a scenario for simulation and to inspect the simulation results. A simulation workflow is described as follows: First, the user forms a grid-based world using tiles of different terrain (cf., Figure 62a). Second, rescue workers of different departments as well as fire places can be positioned on that map. After completing the scenario, the user can choose between two basic mobility models to start the simulation. The sandbox provides a second user interface to visualize the simulation results, similar to a video player software. Finally, a preliminary interface to the peer-to-peer network simulator PlanetSim [74] is also included in this sandbox.



(a) Tile-based world model



(b) OpenStreetMap-based world model

Figure 62: Different world models for disaster simulations

The overall concept was appealing, however, several extensions were required. First, the simulation models described in the previous sections had to be integrated into the simulation framework. This includes all three models: the world model, the mobility model, and the communication model. And second, the sandbox uses a round-based simulation engine to be compatible with the PlanetSim simulator (cf., Section 3.3.3). State of the art network simulators, however, use a discrete-event-based simulation engine. This required a redesign of the simulation engine and a definition of an interface to connect state of the art network simulators. Overall, our goal was to provide an easy way to investigate approaches for service provision including new communication technologies in harsh environments, e.g., in disaster scenarios.

Integration of Simulation Models

We first integrated our OpenStreetMap-based world model described in Section 6.1.1 into the framework. Figure 62 depicts the differences between the tile-based model (Figure 62a) and the OpenStreetMap-based model (Figure 62b). As the tile-based world model is useful

to investigate new and unknown territory (e.g., when planning a new residential area or a new industrial area), the OpenStreetMap-based world model eases the investigation of already existing areas. The OpenStreetMap data can be imported from an OSM-file²² or by selecting a map section online, which is then downloaded directly from the OpenStreetMap servers into our world model. Furthermore, data about the communication infrastructure (e.g., CityMesh data as described in Chapter 4) can be imported from various sources into the world model.

Next, we integrated our mobility and communication models into the framework. Both models interact with the world model and with each other. For instance, the mobility model uses the street network from the underlying map to calculate routes for individual rescue workers as described in Section 6.1.2. Furthermore, messages generated by the communication model (information and orders) can be transmitted by using the communication infrastructure from the world model.

Simulation Engine

In a basic simulation, all messages exchanged between rescue workers are submitted directly from sender to receiver regardless of the underlying transmission medium. This can be interpreted as a strongly simplified abstraction of a conventional analog radio communication used during many of today's rescue missions. Properties of the physical transmission medium are not simulated in this basic simulation. This means that messages are being transmitted instantly and without any message loss.

For a more realistic simulation and to investigate new communication technology, we defined an interface in our framework for the communication model to pass messages on to a network model. At this point, we harness state of the art network simulators that already provide detailed network models. The connected network simulator can then simulate the transmission of messages more realistically including physical properties of the transmission medium. However, this interface required us to redesign the simulation engine of our framework to be compatible with state of the art discrete-event-based network simulators. In such approaches, events are generated for discrete points in time and are ordered and processed chronologically (cf., Section 3.3.3). This allows us to connect our framework with state of the art network simulators for a more fine-grained and realistic disaster simulation. Furthermore, other event-driven simulators can be connected to our simulation framework, as well. A road traffic simulator could simulate traffic in cities, for instance. We present further details on this interlink in Section 6.3.2.

Graphical User Interface

Finally, we improved the quality and performance of the graphical user interface including the visualization of simulation results. DisVis uses OpenGL to smoothly render even large scenarios. Figure 63 shows the visualization module of DisVis with an example simulation result of our hometown Darmstadt. In the top left corner, the hospital is visible and fire fighters are extinguishing a fire on the bottom. Surrounding the fire places, the three disaster zones incident site (red), treatment zone (green), and transport zone (blue) are visible. Also,

²² The OpenStreetMap project specifies an XML-based file format to store their map information in a file.

the CityMesh network is depicted and the encryption of wireless routers is represented by different colors (WPA and WPA2 encryption are depicted in blue, WEP encryption is depicted in yellow, and encryption-less routers are depicted in green). Finally, established wireless network connections are drawn as black lines between routers, rescue workers and vehicles.

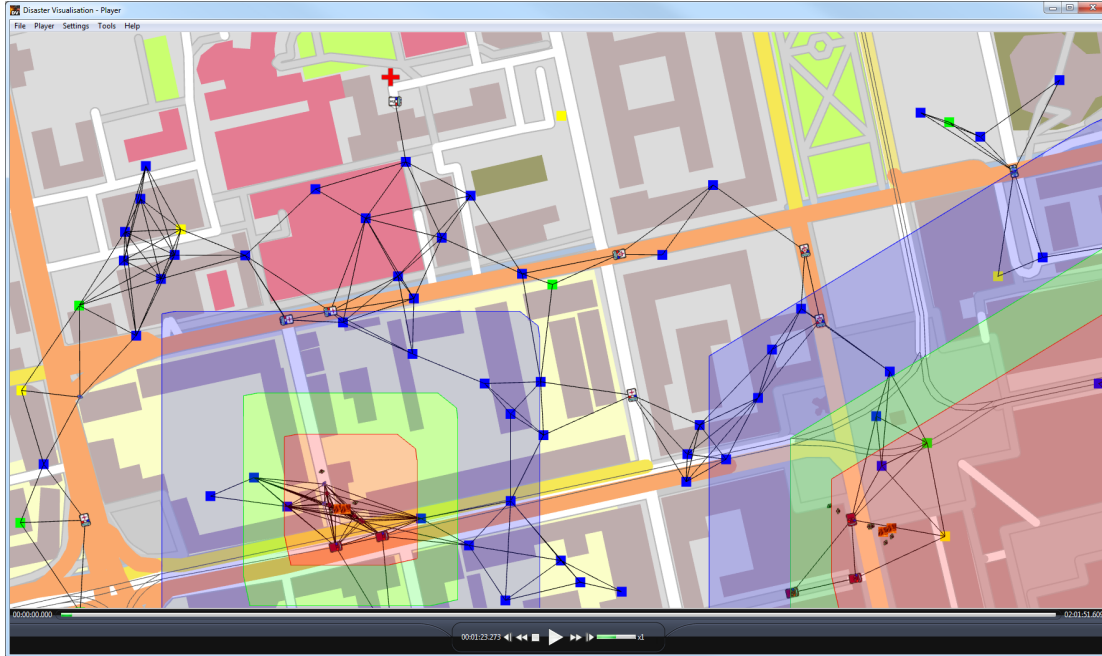


Figure 63: The graphical representation of simulation results

6.3.2 Interlink to Network Simulator

As described above, our simulation framework DisVis is based on a discrete-event-driven simulation engine. Events occur at discrete points in time during the simulation, and the simulation can be described as a chronological sequence of events. The major advantage of such an event-based simulation design is the opportunity to connect our simulation framework with state of the art discrete-event-driven network simulators, such as PeerfactSim.KOM [183], OMNeT++ [192], or ns-3 [85]. The network simulator is then used to simulate different communication technologies in combination with the movement and communication components of our mobility model. Provided the participating nodes react on messages they exchange with one another, their movement and behavior is influenced by the received messages. For example, an ambulance cannot transport a patient to the hospital if it does not receive a command to do so (including the information about the patient's location).

In a conventional state of the art simulation setup, the mobility simulation is executed first, creating movement traces to be imported into the network simulation afterward (cf., Figure 64a). The mobility simulation framework BonnMotion [15], for example, provides many mobility models (including disaster recovery-related models) to create movement traces for various network simulators (cf., Section 3.3.3). In general, movement traces

are static in nature and cannot be adapted dynamically. The movement of simulated participants can, thus, not be changed during a running simulation. We want our system to allow us executing an integrated simulation where network and mobility simulations are executed in parallel. This then provides the added value of an interaction between both simulation models as described in Section 6.1.2. Our integrated simulation concept is depicted in Figure 64b.

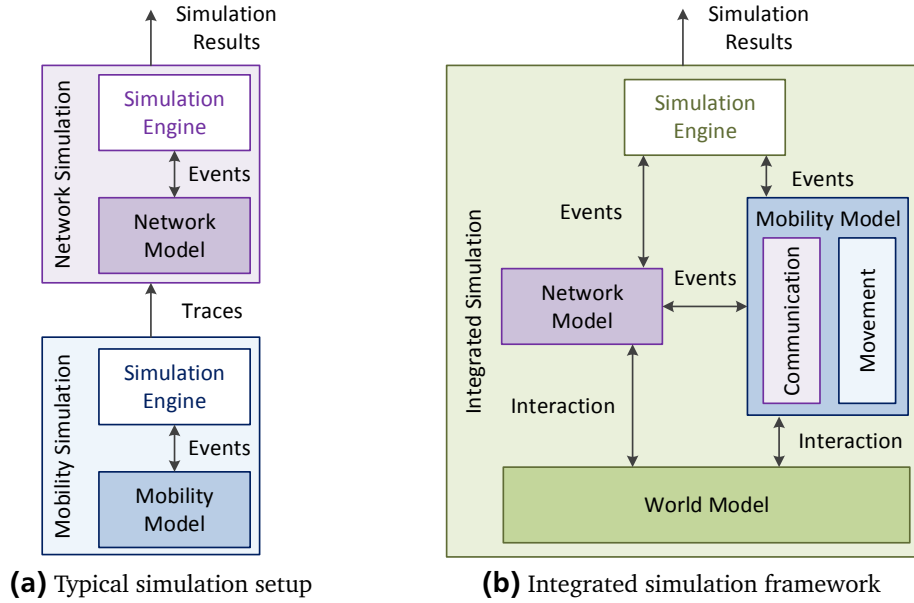


Figure 64: Simulation setups for communications research

To realize such an interactive simulation environment, a shared event-based simulation engine is required to dispatch mobility-related as well as network-related events. This does not necessarily require events to be compatible with both the network and the mobility simulators. Keeping both simulator events separate actually supports openness of the simulator interface. Furthermore, this also enables developers to choose the appropriate network simulator depending on the respective priorities, i.e., high scalability or high level of detail. In our implementation, the events are compatible with both simulators. As we use the state of the art discrete-event-driven network simulator PeerfactSim.KOM [183], we can easily switch between both priorities. It is possible to simulate connections with the IEEE 802.11 WLAN model providing high level of detail or to simulate without this model, thus, improving scalability. Furthermore, we also implemented our peer-to-peer service overlay model in this network simulator. Hence, we do not need to switch to another network simulator for our investigations.

In our integrated simulation concept, the network simulator receives messages generated by the communication component of our mobility model and simulates the message flow between individual network nodes. Arrived messages are then passed back to the mobility model. The messages are evaluated in the mobility model and, according to the implemented rule sets, new movement and communication events are produced. This integrated

simulation framework enables us to investigate the influence of different communication systems on the effectiveness of disaster recovery missions.

6.4 Summary

In this chapter, we presented our approach for the simulation of mobility and communication in disaster recovery missions. We substantiated that, especially in disaster simulations, the interaction and communication between the simulated rescue workers are crucial to the resulting simulation. The reason for this is that most activities are executed by command only, hence, the movement of rescue workers depends on messages received through the communication system. This fact requires that mobility and network simulators are executed together in parallel so that the respective models can exchange information during simulation.

We presented our OpenStreetMap-based simulation framework, called DisVis, with support for detailed agent-based mobility and communication models. It is an event-driven simulation framework capable of stand-alone mobility and communication simulation. To enrich the simulation with realistic network models, we defined an interface to connect our framework with state of the art network simulators. In our concrete implementation, we connected the PeerfactSim.KOM simulator [183] to our framework.

Furthermore, we presented our substantial agent-based mobility model. It defines several agent types for first responders from different rescue departments. Each agent type contains a rule set to define how the respective rescue worker behaves in a certain situation. Our model can be easily extended to simulate various scenarios by implementing new agent types with different rule sets, for instance, a public transportation system in cities.

We also adjusted rules and parameters of our mobility model with information we drew from expert knowledge. We carried out a study based on questionnaires targeted at experts from police departments, fire fighting departments, and medical facilities. Especially medical personnel and fire fighters participated in the surveys. The responses gathered from the surveys allowed us to further improve our rule sets of the different agent types.

7 Conclusion

The presented thesis addresses the multi-faceted problem of *mobile service provision in harsh environments* and particularly in disaster recovery. The main goal of this thesis is to enable rescue workers to consume versatile (cloud) services on top of a resilient communication infrastructure. Achieving this goal helps to improve the effectiveness of first responders during rescue missions.

Harsh environments are defined along the properties of the geographical region, the communication infrastructure, and the participants. The geographical region comprises obstacles (e.g., destroyed buildings and streets) that restrain participants in their movement. The communication infrastructure consists of several stationary devices deployed in the geographical region. In harsh environments, however, parts of the infrastructure are highly unreliable due to destruction or communication overload. Participants are people (e.g., rescue workers) moving through and interacting with the environment. They carry (mobile) devices to interconnect with other participants and to consume versatile services. To achieve the main goal of service provision in such environments, three key challenges have to be faced: (1) improving the resilience of the communication infrastructure, (2) designing a communication system capable of service provision, and (3) conceiving detailed simulation models and tools to investigate such systems in harsh environments.

This thesis went beyond existing concepts and system designs to tackle these challenges. Focusing on the communication infrastructure, this thesis contributed a novel concept to create resilient emergency networks in urban areas by interconnecting private and public wireless routers. Furthermore, this thesis contributed a system architecture for service provision, a novel service placement approach to increase service quality during service provision, and concepts for smooth service migrations. Finally, a detailed mobility model and an integrated simulation framework were contributed by this thesis to comprehensively investigate novel communication systems under real disaster recovery constraints.

In this final chapter, we first summarize the main contributions of this thesis along the three key challenges described above (Section 7.1). Finally, we discuss potential future research directions in Section 7.2.

7.1 Summary of Contributions

We summarize the contributions of the present thesis tackling the key challenges described above as follows:

7.1.1 Communication Infrastructure

The communication infrastructure and the mobile devices of participants (e.g., rescue workers) create the underlay network for service provision. Today, first responders use a fixed communication backbone based on cell type system deployments. Although this infrastructure is readily accessible in case of a disaster, it requires long planning and installation

phases upfront [180]. Furthermore, this communication backbone exhibits single points of failure and can only deliver low data throughput. A decentralized communication infrastructure based on the IEEE 802.11 WLAN standard [90], i.e., a wireless mesh network, would be a much better fit for disaster relief communication [21, 42, 50, 123]. It outperforms existing standards in terms of throughput, openness and failure resilience. However, only few wireless mesh networking projects exist that could be harnessed as communication infrastructure for disaster relief. A more omnipresent infrastructure based on the IEEE 802.11 WLAN standard is yet missing.

In Chapter 4, we contributed a novel approach to create a decentralized emergency communication infrastructure in urban areas. This infrastructure is created by interconnecting private and public wireless routers, thus, forming a wireless mesh network. We propose to integrate an emergency switch into wireless routers. This switch could be triggered by an authorized institution in case of a disaster, thus, creating the communication infrastructure. We call our concept the *CityMesh*.

We evaluated the feasibility of CityMesh networks in three steps: First, we collected data about wireless routers from online sources and with the wardriving technique. Second, we enhanced the wardriving data by using multilateration to estimate the real locations of wireless routers. Third, we constructed networks with the unit disk graph approach and analyzed them for their graph-theoretical properties using the GTNA tool [172]. We showed that public networks in five selected cities differ significantly from randomly generated networks with respect to several graph metrics. The geographical and architectural characteristics of a city (e.g., population density, locations of shops, locations of parks, etc.) play an important role in the distribution of wireless routers throughout the respective city. Furthermore, we found the density of (private and public) wireless routers in the city center of Darmstadt to be very high. If all private and public routers are used to create CityMesh network, a comparably low communication range of only 30 meters is sufficient to create a resilient network that covers the entire city center.

Finally, we discussed several aspects related to the realization of an emergency switch for wireless routers. We pointed out different questions that have to be answered before such emergency switches can be realized. Although our proposed emergency switch for wireless routers might be perceived as a privacy infringement, we believe our concept to be necessary to successfully assist rescue workers in their missions. We, therefore, believe a regulation by the government is required to effectively realize our CityMesh concept.

7.1.2 Peer-to-peer Service Provision

On top of the underlay network that is the communication infrastructure and the mobile devices of participants, a variety of software services can be provided. Although no state of the art concept is able to fulfill the requirements for peer-to-peer-based service provision in harsh environments, the concept of cloudlets [109, 169] exhibits the most interesting properties. Cloudlets are nearby resource-rich computing nodes (e.g., PCs connected to wireless routers in cafés or bars) that could be used for offloading complex computing tasks from the mobile device. We identified service placement to be the crucial component in terms of high quality service provision. State of the art placement approaches do not consider quality of provided services when relocating service instances. Moreover, they aim at reducing the overall network traffic [142]. Furthermore, service placement is based on the code

mobility paradigm [72] and requires smooth service migration to provide interruption-free services. Finally, service placement requires a service discovery approach that supports service mobility. However, state of the art discovery approaches were designed to provide good performance considering peer mobility but regardless of service mobility.

In Chapter 5, we contributed our peer-to-peer service overlay concept. We presented a system model based on the peer-to-peer paradigm and identified necessary system components to fulfill the identified requirements. These components are responsible for service discovery, resource matching, service placement, service replication, and system monitoring, respectively. Furthermore, we conceived four different service placement approaches. Two approaches are based on the quality metric hop count, and the two other approaches are based on the quality metric latency to relocate service instances. The goal of these approaches is to provide a consistent service quality to all connected service consumers. Using a voice communication service, we investigated all four placement approaches in typical disaster recovery settings. We demonstrated that our conceived hop count placement approach is able to achieve consistent service quality by minimizing the hop count for all connected service consumers with only few service migrations and moderate overall message loss. Especially in the command center setup, the hop count placement performed best in terms of consistent hop counts and mouth-to-ear delays for the command center and all group members. In terms of latency-based placement, we came to the conclusion that latency is not a suitable quality metric for service placement.

In our evaluation of six different service discovery approaches, we focused on identifying a suitable service discovery for harsh environments considering service mobility introduced by service placement. Although we selected three state of the art service discovery approaches designed for mobile ad-hoc networks, we found their performance to be disappointing. The naive request flooding approach, however, performed best in terms of discovery performance and data traffic efficiency. Neither peer movement nor frequent service migrations were able to have a drastic negative impact on the discovery performance of this simple approach. Therefore, we recommend the request flooding approach to discover services in peer-to-peer service overlays for harsh environments.

Finally, we contributed four concepts for smooth service migration: soft handover, persistent socket connections, efficient serialization, and efficient network protocol. We presented our real-world experiment setup, called *PeerMoS* to investigate smooth service migration and demonstrated that bandwidth is a rare resource in wireless multi-hop networks. Hence, service migration must not only be smooth but also efficient in terms of bandwidth consumption. The soft handover concept and the persistent socket connections aim at eliminating the interruptions and service unavailability during service migration. The efficient serialization and the efficient network protocol aim at reducing data traffic generated by our peer-to-peer service overlay prototype.

7.1.3 Simulation Models and Tools for Harsh Environments

To evaluate the effectiveness of new communication systems (i.e., our proposed peer-to-peer service overlay concept) in realistic harsh environments, a detailed world model and an integrated mobility and communication model are required. In disaster relief, the movement of rescue workers depends on various factors: the terrain and the current disaster situation, the mission strategy of the respective rescue workers, and the orders received

from the command centers. However, none of the state of the art mobility models integrates all these features. A combination of the zone-based [16], role-based [136], and gravity-based [136] mobility models would produce the most realistic movement patterns for disaster relief simulations. Although the BonnMotion [15] framework provides suitable disaster-related mobility models for abstract simulations, an integrated framework that provides versatile mobility models for disaster relief and a realistic WLAN model is yet missing.

In Chapter 6, we contributed an integrated simulation framework, called *DisVis*, to investigate communication systems in realistic harsh environments. It builds on an OpenStreetMap-based world model and supports detailed agent-based mobility models. It is an event-driven simulation framework capable of stand-alone mobility and communication simulation. We substantiated that, especially in disaster simulations, the communication and interaction between the simulated rescue workers are crucial to generate meaningful results. The reason for this is that most activities are executed by command only. Hence, the movement of rescue workers depends on messages received through the communication system. This fact requires that mobility and network simulators are executed together in parallel so that the respective models can exchange information during simulation. To enrich the simulation with realistic network models, we, therefore, defined an interface to connect our framework with state of the art network simulators. In our implementation, we connected the PeerfactSim.KOM simulator [183] to our framework.

Furthermore, we presented our substantial agent-based mobility model. It defines several agent types for first responders from different rescue departments. Each agent type contains a rule set to define how the respective rescue workers behave in certain situations. Our model can be easily extended to simulate various scenarios by implementing new agent types with different rule sets, for instance, a public transportation system in cities. We further adjusted rules and parameters of our mobility model with results from our survey targeted at experts from police departments, fire fighting departments, and medical facilities.

7.2 Potential Directions for Future Research

In the evaluation of our CityMesh concept, we showed that networks in five selected cities differ significantly from randomly generated networks with respect to several graph metrics. The geographical and architectural characteristics of a city (e.g., population density, locations of shops, locations of parks, etc.) play an important role in the distribution of wireless routers throughout the respective city. For network simulations, however, it is typically assumed that node placement follows a uniform distribution. Although the existing work NPART [127] relies on realistic but comparably small data sets about the Freifunk networks in Leipzig and Berlin, it does not consider some relevant metrics, e.g., cluster coefficients. In addition, evaluated metrics comprise significantly different values for both data sets. Based on our results and NPART, a consequent next step would be to develop a more realistic node placement algorithm for simulations of wireless routers in urban areas.

Furthermore, we highlighted that security of our proposed CityMesh concept is a big issue. We argued that most security-related issues could be resolved with state of the art approaches and that a physical separation of the emergency and home networks would further increase the obstacle for attackers to compromise the system. Overall, we believe that the emergency switch for wireless routers is the most critical aspect in our concept. We

identified two possibilities to activate the switch: via Internet and via a wireless beacon sent out by first responders. The Internet-based activation scheme relies on Internet connectivity, which cannot be assumed during disaster recovery. Hence, the remaining question is: *How could a beacon-based activation scheme be secured from unauthorized activation?*

Our smooth service migration concept relies on soft handover. This scheme replicates a service instance onto another peer, synchronizes execution states, and shuts the old service instance down as soon as the new service instance is ready to take over service provision. If the old service instance would not be shut down but instead kept alive and synchronized, this service instance could take over in case the primary instance fails. On the other hand, replicating the service instance in the direct neighborhood could improve performance of service placement since no code migration would be needed. A take over message would then simply switch service provision to another active service instance. We, therefore, formulate two research questions based on replication and placement: *Can service placement make the additional replication component obsolete? Should service placement and replication be combined to improve the overall system performance?*

Furthermore, we investigated several state of the art approaches to be implemented as the identified system components. Most of these approaches provide high performance under certain constraints. Once these constraints change, they tend to fail. We have seen this also for the service discovery. As the service overlay in harsh environments is exposed to highly dynamic constraints by the environment, we believe that system components should dynamically adapt to these constraints. This could either happen by adjusting the currently active system component or by replacing it with another implementation. Hence, we formulate the following research question: *How to transit between different approaches for individual system components during runtime to improve the overall system performance?*

Finally, we presented concepts to investigate the impact of different communication systems on the effectiveness of disaster recovery missions. With our provided models and tools, existing and future communication systems can now be evaluated for their applicability to harsh environments.



Bibliography

- [1] M. Abolhasan, B. Hagelstein, and J. C.-P. Wang, “Real-world Performance of Current Proactive Multi-hop Mesh Protocols,” in *Proceedings of the 15th Asia-Pacific Conference on Communications*, no. Apcc. IEEE, Oct 2009, pp. 44–47.
- [2] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, “A Review of Routing Protocols for Mobile Ad Hoc Networks,” *Ad Hoc Networks*, vol. 2, no. 1, pp. 1–22, Jan. 2004.
- [3] L. Abusalah, A. Khokhar, and M. Guizani, “A Survey of Secure Mobile Ad Hoc Routing Protocols,” *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 78–93, 2008.
- [4] M. Afanasyev, T. Chen, G. M. Voelker, and A. C. Snoeren, “Analysis of a Mixed-use Urban Wifi Network: When Metropolitan Becomes Neapolitan,” in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: ACM, 2008, pp. 85–98.
- [5] —, “Usage Patterns in an Urban WiFi Network,” *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1359–1372, Oct. 2010.
- [6] D. Aguayo, J. Bicket, S. Biswas, and D. De Couto, “MIT Roofnet: Construction of a Production Quality Ad-Hoc Network,” in *Poster at the International Conference on Mobile Computing and Networking*, 2003.
- [7] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser, “MundoCore: A Light-weight Infrastructure for Pervasive Computing,” *Pervasive and Mobile Computing*, vol. 3, no. 4, pp. 332 – 361, 2007.
- [8] K. Akkaya and M. Younis, “A Survey on Routing Protocols for Wireless Sensor Networks,” *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, May 2005.
- [9] I. F. Akyildiz, X. Wang, and W. Wang, “Wireless Mesh Networks: A survey,” *Computer Networks*, vol. 47, no. 4, pp. 445–487, Mar. 2005.
- [10] I. Akyildiz and X. Wang, “A Survey on Wireless Mesh Networks,” *IEEE Communications Magazine*, vol. 43, no. 9, pp. S23–S30, Sep. 2005.
- [11] S. Ali, A. Mitschele-Thiel, A. Diab, and A. Rasheed, “A Survey of Services Placement Mechanisms for Future Mobile Communication Networks,” in *Proceedings of the 8th International Conference on Frontiers of Information Technology*. New York, NY, USA: ACM, 2010, pp. 39:1–39:5.
- [12] T. Amjad, M. Sher, and A. Daud, “A Survey of Dynamic Replication Strategies for Improving Data Availability in Data Grids,” *Future Generation Computer Systems*, vol. 28, no. 2, pp. 337 – 349, 2012.

-
- [13] A. Arjona and S. Takala, "The Google Muni Wifi Network—Can it Compete with Cellular Voice?" in *Proceedings of the Third Advanced International Conference on Telecommunications*. IEEE, May 2007, pp. 11–16.
- [14] K. Arnold, R. Scheifler, J. Waldo, B. O'Sullivan, and A. Wollrath, *Jini Specification*, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.
- [15] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "BonnMotion: A Mobility Scenario Generation and Analysis Tool," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. Brussels, Belgium: ICST, 2010, pp. 51:1–51:10.
- [16] N. Aschenbruck, E. Gerhards-Padilla, and P. Martini, "Modeling Mobility in Disaster Area Scenarios," *Performance Evaluation*, vol. 66, no. 12, pp. 773–790, 2009.
- [17] N. Aschenbruck, M. Gerharz, M. Frank, and P. Martini, "Modelling Voice Communication in Disaster Area Scenarios," *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pp. 211–220, Nov. 2006.
- [18] N. Aschenbruck and P. Martini, "Evaluation and parameterization of voice traffic models for disaster area scenarios," in *Proceedings of the 33rd IEEE Conference on Local Computer Networks*. IEEE, Oct 2008, pp. 236–243.
- [19] N. Aschenbruck, P. Martini, and M. Gerharz, "Characterisation and Modelling of Voice Traffic in First Responder Networks," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks*, Oct. 2007, pp. 295–302.
- [20] N. Aschenbruck, A. Munjal, and T. Camp, "Trace-based Mobility Modeling for Multi-hop Wireless Networks," *Computer Communications*, vol. 34, no. 6, pp. 704–714, May 2011.
- [21] A. Bahora, T. Collins, S. Davis, S. Goknur, J. Kearns, T. Lieu, T. Nguyen, J. Zeng, B. Horowitz, and S. Patek, "Integrated Peer-to-peer Applications for Advanced Emergency Response Systems. Part I. Concept of Operations," in *Proceedings of the IEEE Systems and Information Engineering Design Symposium*, April 2003, pp. 255–260.
- [22] F. Bai and A. Helmy, "IMPORTANT: A Framework to Systematically Analyze the Impact of Mobility on Performance of Routing Protocols for Adhoc Networks," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, March 2003, pp. 825–835.
- [23] J. Balasubramanian, S. Tambe, C. Lu, A. Gokhale, C. Gill, and D. C. Schmidt, "Adaptive Failover for Real-Time Middleware with Passive Replication," in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2009, pp. 118–127.
- [24] M. Balmer, N. Cetin, K. Nagel, and B. Raney, "Towards Truly Agent-Based Traffic and Mobility Simulations," in *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, July 2004, pp. 60–67.

-
- [25] J. Q. Bao and W. C. Lee, "Rapid Deployment of Wireless Ad Hoc Backbone Networks for Public Safety Incident Management," in *Proceedings of the IEEE Global Telecommunications Conference*, Nov 2007, pp. 1217–1221.
- [26] M. Barbeau, "Point-to-point Voice over Ad Hoc Networks: A Survey," *Pervasive and Mobile Computing*, vol. 8, no. 3, pp. 376–387, Jun. 2012.
- [27] F. Barcelo and S. Bueno, "Idle and Inter-arrival Time Statistics in Public Access Mobile Radio (PAMR) Systems," in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 1. IEEE, Nov 1997, pp. 126–130.
- [28] R. Barr, Z. J. Haas, and R. van Renesse, "JiST: An Efficient Approach to Simulation Using Virtual Machines," *Software: Practice and Experience*, vol. 35, no. 6, pp. 539–576, 2005.
- [29] P. Bellavista, A. Corradi, and E. Magistretti, "REDMAN: An Optimistic Replication Middleware for Read-only Resources in Dense MANETs," *Pervasive and Mobile Computing*, vol. 1, no. 3, pp. 279–310, Sep. 2005.
- [30] K. A. Bharat and L. Cardelli, "Migratory Applications," in *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. New York, NY, USA: ACM, 1995, pp. 132–142.
- [31] S. Biswas and R. Morris, "ExOR: Opportunistic Multi-hop Routing for Wireless Networks," *SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 133–144, Aug. 2005.
- [32] T. Bönning, "Integration von Suche und Platzierung mobiler Dienste in Service Overlays," Telecooperation, Technische Universität Darmstadt, Master's Thesis, 2013.
- [33] D. Bradler, "Peer-to-Peer Concepts for Emergency First Response," Ph.D. dissertation, Fachbereich Informatik, Technische Universität Darmstadt, Juni 2010.
- [34] D. Bradler, J. Kangasharju, and M. Mühlhäuser, "Evaluation of Peer-to-Peer Overlays for First Response," in *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications*, Mar. 2008, pp. 463–467.
- [35] D. Bradler, B. Schiller, E. Aitenbichler, and N. Liebau, "Towards a Distributed Crisis Response Communication System," in *Proceedings of the 6th International Conference on Information Systems for Crisis Response and Management*, no. May, 2009.
- [36] D. Bradler, I. Schweizer, K. Panitzek, and M. Mühlhäuser, "First Response Communication Sandbox," in *Proceedings of the 11th Communications and Networking Simulation Symposium*. New York, NY, USA: ACM, 2008, pp. 115–122.
- [37] W. Bradley and D. Maher, "The NEMO P2P Service Orchestration Framework," in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*. IEEE, Jan 2004, pp. 10 pp.–.
- [38] P. T. Brady, "A Model for Generating On-Off Speech Patterns in Two-Way Conversation," *Bell System Technical Journal*, vol. 48, no. 7, pp. 2445–2472, Sep. 1969.

-
- [39] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.
- [40] D. Caromel, C. Delbe, A. Di Costanzo, and M. Leyton, "ProActive: An Integrated Platform for Programming and Running Applications on Grids and P2P Systems," *Computational Methods in Science and Technology*, vol. 12, no. 1, 2006.
- [41] D. Caruso, "Vergleich unterschiedlicher Kommunikationstechnologien in Katastropheneinsätzen," Telecooperation, Technische Universität Darmstadt, Bachelor's Thesis, 2014.
- [42] T. Catarci, M. D. Leoni, A. Marrella, M. Mecella, B. Salvatore, G. Vetere, S. Dustdar, L. Juszczak, A. Manzoor, and H.-L. Truong, "Pervasive Software Environments for Supporting Disaster Responses," *IEEE Internet Computing*, vol. 12, no. 1, pp. 26–37, Jan. 2008.
- [43] I. Chakeres and C. Perkins, "Dynamic MANET On-demand (DYMO) routing," *Internet-Draft: draft-ietf-manet-dymo-14 (work in progress)*, pp. 1–37, 2010.
- [44] D. Chakraborty, A. Joshi, Y. Yesha, and T. Finin, "Toward Distributed service discovery in pervasive computing environments," *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 97–112, Feb. 2006.
- [45] —, "GSD: A Novel Group-based Service Discovery Protocol for MANETS," in *Proceedings of the 4th International Workshop on Mobile and Wireless Communications Network*. IEEE, 2002, pp. 140–144.
- [46] A. Chakravarti, G. Baumgartner, and M. Lauria, "The Organic Grid: Self-Organizing Computation on a Peer-to-Peer Network," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 35, no. 3, pp. 373–384, May 2005.
- [47] B. A. Chambers, "The Grid Roofnet: A Rooftop Ad Hoc Wireless Network," Masters Thesis, Massachusetts Institute of Technology, 2002.
- [48] L.-H. Chang, C.-H. Sung, H.-C. Chu, and J.-J. Liaw, "Design and Implementation of the Push-to-talk Service in Ad Hoc VoIP Network," *IET Communications*, vol. 3, no. 5, pp. 740–751, May 2009.
- [49] G. Chen, C. P. Low, and Z. Yang, "Coordinated Services Provision in Peer-to-Peer Environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 4, pp. 433–446, Apr. 2008.
- [50] R. Chen, R. Sharman, H. Rao, and S. Upadhyaya, "Design Principles of Coordinated Multi-incident Emergency Response Systems," in *Intelligence and Security Informatics*, ser. Lecture Notes in Computer Science, P. Kantor, G. Muresan, F. Roberts, D. Zeng, F.-Y. Wang, H. Chen, and R. Merkle, Eds. Springer Berlin Heidelberg, 2005, vol. 3495, pp. 81–98.
- [51] J. Chroboczek, "RFC 6126 (Experimental) - The Babel Routing Protocol," *Internet RFCs*, pp. 1–45, 2011.

-
- [52] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution Between Mobile Device and Cloud," in *Proceedings of the Sixth Conference on Computer Systems*. New York, NY, USA: ACM, 2011, pp. 301–314.
- [53] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs," *Discrete Mathematics*, vol. 86, no. 1-3, pp. 165–177, Dec. 1990.
- [54] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," in *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286.
- [55] T. Clausen and P. Jacquet, "RFC 3626 (Experimental) - Optimized Link State Routing Protocol (OLSR)," *Internet RFCs*, pp. 1–75, 2003.
- [56] A. County, "After-action Report on the Response to the September 11 Terrorist Attack on the Pentagon," *Prepared by Titan Systems Corp*, 2002.
- [57] R. Cover, "Universal Description, Discovery, and Integration (UDDI)," 2008, visited on April 4, 2014. [Online]. Available: <http://xml.coverpages.org/uddi.html>
- [58] E. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
- [59] R. Dilmaghani and R. Rao, "A Wireless Mesh Infrastructure Deployment with Application for Emergency Scenarios," in *Proceedings of the 5th International Conference on Information Systems for Crisis Response and Management*, 2008, pp. 1–11.
- [60] W. Effelsberg, R. Steinmetz, and T. Strufe, Eds., *Benchmarking Peer-to-Peer Systems – Understanding Quality of Service in Large-Scale Distributed Systems*, 7847th ed., ser. LNCS State of the Art Surveys. Springer, 2013.
- [61] Y.-H. Eom, K.-D. Jung, and Y.-K. Choi, "On-line Game Service System Using Grid Technology Based on P2P Environment," in *Proceedings of the 1st International Conference on Scalable Information Systems*. New York, NY, USA: ACM, 2006.
- [62] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "Epidemic Information Dissemination in Distributed Systems," *IEEE Computer*, vol. 37, no. 5, pp. 60–67, May 2004.
- [63] D. Faria, "Modeling Signal Attenuation in IEEE 802.11 Wireless LANs-vol. 1," *Computer Science Department, Stanford University*, vol. 1, 2005.
- [64] M. Fiore, J. Harri, F. Filali, and C. Bonnet, "Vehicular Mobility Simulation for VANETs," in *Proceedings of the 40th Annual Simulation Symposium*. New York, NY, USA: IEEE, March 2007, pp. 301–309.
- [65] D. Florencio and C. Herley, "A Large-scale Study of Web Password Habits," in *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA: ACM, 2007, pp. 657–666.

-
-
- [66] H. Flores and S. N. Srirama, "Mobile Cloud Middleware," *Journal of Systems and Software*, vol. 92, no. 0, pp. 82–94, Jun. 2014.
- [67] H. Flores, S. N. Srirama, and C. Paniagua, "Towards Mobile Cloud Applications: Offloading Resource-intensive Tasks to Hybrid Clouds," *International Journal of Pervasive Computing and Communications*, vol. 8, no. 4, pp. 344–367, 2012.
- [68] I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," in *Peer-to-Peer Systems II*, ser. Lecture Notes in Computer Science, M. Kaashoek and I. Stoica, Eds. Springer Berlin Heidelberg, 2003, vol. 2735, pp. 118–128.
- [69] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *Proceedings of the Grid Computing Environments Workshop*. IEEE, Nov 2008, pp. 1–10.
- [70] C. Frank and K. Römer, "Distributed Facility Location Algorithms for Flexible Configuration of Wireless Sensor Networks," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science, J. Aspnes, C. Scheideler, A. Arora, and S. Madden, Eds. Springer Berlin Heidelberg, 2007, vol. 4549, pp. 124–141.
- [71] L. Fritsche, "Aktivierung von Ad-hoc und Mesh Funktionalität auf Android Smartphones," Telecooperation, Technische Universität Darmstadt, Bachelor's Thesis, 2013.
- [72] A. Fuggetta, G. Picco, and G. Vigna, "Understanding Code Mobility," *IEEE Transactions on Software Engineering*, vol. 24, no. 5, pp. 342–361, May 1998.
- [73] Z.-g. Gao, X.-z. Yang, T.-y. Ma, and S.-b. Cai, "RICFFP: An Efficient Service Discovery Protocol for MANETs," in *Embedded and Ubiquitous Computing*, ser. Lecture Notes in Computer Science, L. Yang, M. Guo, G. Gao, and N. Jha, Eds. Springer Berlin Heidelberg, 2004, vol. 3207, pp. 786–795.
- [74] P. García, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo, "PlanetSim: A New Overlay Network Simulation Framework," pp. 123–136, 2005.
- [75] K. Graffi, D. Stingl, C. Gross, H. Nguyen, A. Kovacevic, and R. Steinmetz, "Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems," in *Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems*, Dec 2010, pp. 27–34.
- [76] X. Gu, K. Nahrstedt, and B. Yu, "SpiderNet: An Integrated Peer-to-Peer Service Composition Framework," in *Proceedings of the 13th IEEE International Symposium on High performance Distributed Computing*, June 2004, pp. 110–119.
- [77] R. Guerin, "Channel Occupancy Time Distribution in a Cellular Radio System," *IEEE Transactions on Vehicular Technology*, vol. 36, no. 3, pp. 89–99, Aug. 1987.
- [78] Z. Haas, "A New Routing Protocol for the Reconfigurable Wireless Networks," in *Proceedings of 6th IEEE International Conference on Universal Personal Communications Record*, vol. 2. IEEE, Oct 1997, pp. 562–566.

-
-
- [79] Z. J. Haas, M. R. Pearlman, and P. Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," *Internet-Draft: draft-ietf-manet-zone-zrp-04*, pp. 1–11, 2002.
- [80] M. M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, Oct 2008.
- [81] F. Hammacher, "Optimierung von Dienstmigration und Kommunikation in Peer-to-peer Service Overlays," Telecooperation, Technische Universität Darmstadt, Bachelor's Thesis, 2013.
- [82] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "VanetMobiSim: Generating Realistic Mobility Patterns for VANETs," in *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*. New York, NY, USA: ACM, 2006, pp. 96–97.
- [83] N. Hegde, A. Proutiere, and J. Roberts, "Evaluating the Voice Capacity of 802.11 WLAN under Distributed Control," in *Proceedings of the 14th IEEE Workshop on Local & Metropolitan Area Networks*. IEEE, Sept 2005, pp. 1–6.
- [84] S. Helal, N. Desai, V. Verma, and C. Lee, "Konark - A Service Discovery and Delivery Protocol for Ad-hoc Networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference*, vol. 3, no. C, March 2003, pp. 2107–2113.
- [85] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "Ns-3 Project Goals," in *Proceeding from the 2006 Workshop on Ns-2: The IP Network Simulator*. New York, NY, USA: ACM, 2006.
- [86] G. R. Hiertz, S. Max, R. Zhao, D. Denteneer, and L. Berlemann, "Principles of IEEE 802.11s," in *Proceedings of 16th International Conference on Computer Communications and Networks*. IEEE, Aug. 2007, pp. 1002–1007.
- [87] X. Hong, M. Gerla, G. Pei, and C.-c. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," in *Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. New York, NY, USA: ACM, 1999, pp. 53–60.
- [88] A. W. Howe, M. E. Jennex, G. H. Bressler, and E. Frost, "Exercise 24: Using Social Media for Crisis Response," *International Journal of Information Systems for Crisis Response and Management*, vol. 3, no. 4, pp. 36–54, 2011.
- [89] G. Huppertz, "Chorus – Car Horns Used as Sirens," in *Fraunhofer Symposium Future Security*, 2010, pp. 294–297).
- [90] IEEE, "IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.
- [91] ITU-T, "Recommendation G.711 – Pulse code modulation (PCM) of voice frequencies," 1988.

-
- [92] —, “P59 – Artificial Conversational Speech,” 1993.
- [93] —, “Recommendation G.114 – One-way transmission time,” 2003.
- [94] —, “Recommendation G.723.1 – Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s,” 2006.
- [95] —, “Recommendation G.729 – Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP),” 2012.
- [96] M. Jelasity, A. Montresor, and O. Babaoglu, “Gossip-based Aggregation in Large Dynamic Networks,” *ACM Transactions on Computer Systems*, vol. 23, no. 3, pp. 219–252, Aug. 2005.
- [97] M. E. Jennex, “Social Media – Viable for Crisis Response? Experience from the Great San Diego/Southwest Blackout,” *International Journal of Information Systems for Crisis Response and Management*, vol. 4, no. 2, pp. 53–67, 2012.
- [98] D. Johnson, Y. Hu, and D. Maltz, “RFC 4728 (Experimental) - The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4,” *Internet RFCs*, pp. 1–107, 2007.
- [99] D. Johnson and D. Maltz, “Dynamic Source Routing in Ad Hoc Wireless Networks,” in *Mobile Computing*, ser. The Kluwer International Series in Engineering and Computer Science, T. Imielinski and H. Korth, Eds. Springer US, 1996, vol. 353, pp. 153–181.
- [100] J. Jordan and F. Barcelo, “Statistical Modelling of Transmission Holding Time in PAMR Systems,” in *Proceedings of the IEEE Global Telecommunications Conference*, vol. 1. IEEE, Nov 1997, pp. 121–125.
- [101] L. Junhai, Y. Danxia, X. Liu, and F. Mingyu, “A Survey of Multicast Routing Protocols for Mobile Ad-Hoc Networks,” *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 78–91, 2009.
- [102] K. Kanchanasut, A. Tunpan, M. A. Awal, D. K. Das, T. Wongsardsakul, and Y. Tsuchimoto, “DUMBONET: A Multimedia Communication System for Collaborative Emergency Response Operations in Disaster-affected Areas,” *International Journal of Emergency Management*, vol. 4, no. 4, pp. 670–681, 2007.
- [103] W. Kellerer, Z. Despotovic, M. Michel, Q. Hofstatter, and S. Zols, “Towards a Mobile Peer-to-Peer Service Platform,” in *Proceedings of the International Symposium on Applications and the Internet Workshops*. IEEE, Jan 2007, pp. 2–2.
- [104] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based Computation of Aggregate Information,” in *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, Oct 2003, pp. 482–491.
- [105] A. Keränen, J. Ott, and T. Kärkkäinen, “The ONE Simulator for DTN Protocol Evaluation,” in *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques*. ICST, Brussels, Belgium: ICST, 2009, pp. 55:1–55:10.

-
- [106] A. Kipnis and B. Patt-Shamir, "A Note on Distributed Stable Matching," in *Proceedings of the 29th IEEE International Conference on Distributed Computing Systems*, Jun. 2009, pp. 466–473.
- [107] M. Klein, B. König-Ries, and P. Obreiter, "Service Rings - A Semantic Overlay for Service Discovery in Ad Hoc Networks," in *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*. IEEE, Sept 2003, pp. 180–185.
- [108] M. Klein and B. König-Ries, "Multi-layer clusters in ad-hoc networks — an approach to service discovery," in *Web Engineering and Peer-to-Peer Computing*, ser. Lecture Notes in Computer Science, E. Gregori, L. Cherkasova, G. Cugola, F. Panzieri, and G. Picco, Eds. Springer Berlin Heidelberg, 2002, vol. 2376, pp. 187–201.
- [109] E. Koukoulidis, D. Lymberopoulos, K. Strauss, J. Liu, and D. Burger, "Pocket Cloudlets," in *Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: ACM, 2011, pp. 171–184.
- [110] U. C. Kozat and L. Tassiulas, "Service Discovery in Mobile Ad Hoc Networks: An Overall Perspective on Architectural Choices and Network Layer Support Issues," *Ad Hoc Networks*, vol. 2, no. 1, pp. 23–44, Jan. 2004.
- [111] D. Krajzewicz, G. Hertkorn, P. Wagner, and C. Rössel, "SUMO (Simulation of Urban MObility) An Open-source Traffic Simulation," in *Proceedings of the 4th Middle East Symposium on Simulation and Modelling*, 2002, pp. 183–187.
- [112] S. Kurkowski, T. Camp, and M. Colagrosso, "MANET Simulation Studies: The Incredibles," *SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 4, pp. 50–61, Oct. 2005.
- [113] M. Lamanna, "The LHC Computing Grid Project at CERN," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 534, no. 1-2, pp. 1–6, Nov. 2004.
- [114] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed Placement of Service Facilities in Large-Scale Networks," in *Proceedings of the 26th IEEE International Conference on Computer Communications*. IEEE, May 2007, pp. 2144–2152.
- [115] H. Lee, "Online Stable Matching as a Means of Allocating Distributed Resources," *Journal of Systems Architecture*, vol. 45, no. 15, pp. 1345–1355, Sep. 1999.
- [116] H. Liu, T. Roeder, K. Walsh, R. Barr, and E. G. Sirer, "Design and Implementation of a Single System Image Operating System for Ad Hoc Networks," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA: ACM, 2005, pp. 149–162.
- [117] M. Liu, T. Koskela, Z. Ou, J. Zhou, J. Riekk, and M. Ylianttila, "Super-peer-based Coordinated Service Provision," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1210–1224, Jul. 2011.

-
-
- [118] K. Loesing and S. Kaffile, “Open Chord Project Homepage,” 2011, visited on March 12, 2014. [Online]. Available: <http://www.sourceforge.net/projects/open-chord>
- [119] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “TAG: A Tiny AGgregation Service for Ad-Hoc Sensor Networks,” *SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, Dec. 2002.
- [120] V. March, Y. Gu, E. Leonardi, G. Goh, M. Kirchberg, and B. S. Lee, “ μ Cloud: Towards a New Paradigm of Rich Mobile Applications,” *Procedia Computer Science*, vol. 5, no. 0, pp. 618–624, Jan. 2011.
- [121] P. Maymounkov and D. Mazières, “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,” in *Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Springer Berlin Heidelberg, 2002, vol. 2429, pp. 53–65.
- [122] S. McCanne, S. Floyd, K. Fall, K. Varadhan *et al.*, “The network simulator ns-2,” 1997, visited on May 1, 2014. [Online]. Available: <http://www.isi.edu/nsnam/ns/>
- [123] M. Mecella, M. Angelaccio, A. Krek, T. Catarci, B. Buttarazzi, and S. Dustdar, “WORK-PAD: An Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios,” in *Proceedings of the International Symposium on Collaborative Technologies and Systems*. IEEE, May 2006, pp. 173–180.
- [124] A. N. Mian, R. Baldoni, and R. Beraldi, “A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks,” *IEEE Pervasive Computing*, vol. 8, no. 1, pp. 66–74, Jan. 2009.
- [125] B. Milic and M. Malek, “Adaptation of the Breadth First Search Algorithm for Cut-edge Detection in Wireless Multihop Networks,” in *Proceedings of the 10th ACM Symposium on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*. New York, NY, USA: ACM, 2007, pp. 377–386.
- [126] —, “Analyzing Large Scale Real-World Wireless Multihop Network,” *IEEE Communications Letters*, vol. 11, no. 7, pp. 580–582, Jul. 2007.
- [127] —, “NPART - Node Placement Algorithm for Realistic Topologies in Wireless Multihop Network Simulation,” in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*. Brussels, Belgium: ICST, 2009, pp. 9:1–9:10.
- [128] T. Mitschke, *Handbuch für Schnell-Einsatz-Gruppen*. Stumpf & Kossendey, 1994.
- [129] L. Morales, “Home internet access still out of reach for many world-wide,” Website, 2013, available online at <http://www.gallup.com/poll/159815/home-internet-access-remains-reachworldwide.aspx>; visited on May 1, 2014.
- [130] T. Moscibroda and R. Wattenhofer, “Facility Location: Distributed Approximation,” in *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing*. New York, NY, USA: ACM, 2005, pp. 108–117.

-
- [131] S. Motegi, K. Yoshihara, and H. Horiuchi, "Service Discovery for Wireless Ad Hoc Networks," in *Proceedings of the 5th International Symposium on Wireless Personal Multimedia Communications*, vol. 1. IEEE, Oct 2002, pp. 232–236.
- [132] S. M. Mousavi, H. R. Rabiee, M. Moshref, and A. Dabirmoghaddam, "MobiSim: A Framework for Simulation of Mobility Models in Mobile Ad-Hoc Networks," in *Proceedings of the International Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE, Oct 2007, pp. 82–82.
- [133] D. Murray, M. Dixon, and T. Koziniec, "An Experimental Comparison of Routing Protocols in Multi Hop Ad Hoc Networks," in *Proceedings of the Australasian Telecommunication Networks and Applications Conference*. IEEE, Oct 2010, pp. 159–164.
- [134] S. Naicken, A. Basu, B. Livingston, and S. Rodhetbhai, "A Survey of Peer-to-Peer Network Simulators," in *Proceedings of The Seventh Annual Postgraduate Symposium*, vol. 2, 2006.
- [135] A. Nedos, K. Singh, and S. Clarke, "Service*: Distributed Service Advertisement for Multi-service, Multi-hop MANET Environments," in *Proceedings of the 7th IFIP International Conference on Mobile and Wireless Communication Networks*, 2005.
- [136] S. C. Nelson, A. F. Harris, and R. Kravets, "Event-driven, Role-based Mobility in Disaster Recovery Networks," in *Proceedings of the Second ACM Workshop on Challenged Networks*. New York, NY, USA: ACM, 2007, pp. 27–34.
- [137] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)," *Internet-Draft: draft-wunderlich-openmesh-manet-routing-00*, pp. 1–75, 2008.
- [138] D. Niculescu, S. Ganguly, K. Kim, and R. Izmailov, "Performance of VoIP in a 802.11 Wireless Mesh Network," in *Proceedings of the 25th IEEE International Conference on Computer Communications*. IEEE, April 2006, pp. 1–11.
- [139] M. Nidd, "Service Discovery in DEAPspace," *IEEE Personal Communications*, vol. 8, no. 4, pp. 39–45, Aug 2001.
- [140] Office for Interoperability and Compatibility Department of Homeland Security, "Public Safety Statement of Requirements for Communications & Interoperability," *The SAFECOM Program*, vol. 2, 2008.
- [141] K. Oikonomou and I. Stavrakakis, "Scalable Service Migration: The Tree Topology," in *Proceedings of the 5th Annual Mediterranean Ad Hoc Networking Workshop*, Lipari, Italy, 2006, pp. 1–8.
- [142] —, "Scalable Service Migration in Autonomic Network Environments," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 1, pp. 84–94, Jan. 2010.
- [143] K. Oikonomou, I. Stavrakakis, and A. Xydias, "Scalable Service Migration in General Topologies," in *Proceedings of the 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, Jun. 2008, pp. 1–6.

-
-
- [144] T. Okoshi, M. Mochizuki, Y. Tobe, and H. Tokuda, "MobileSocket: Toward Continuous Operation for Java Applications," in *Proceedings of the 8th International Conference on Computer Communications and Networks*, 1999, pp. 50–57.
- [145] A. G. on Non-ionising Radiation, "Possible Health Effects from Terrestrial Trunked Radio (TETRA)," *Report of an advisory group on nonionising radiation documents of the NRPB*, vol. 12, pp. 1–40, 2001.
- [146] F. A. Onat, I. Stojmenovic, and H. Yanikomeroglu, "Generating Random Graphs for the Simulation of Wireless Ad Hoc, Actuator, Sensor, and Internet Networks," *Pervasive and Mobile Computing*, vol. 4, no. 5, pp. 597–615, Oct. 2008.
- [147] K. Panitzek, "Concepts for Service Provision in Disaster Environments," in *Proceedings of the International Conference on Pervasive Computing and Communications Workshops*. Lugano, Switzerland: IEEE, Mar 2012, pp. 558 – 559.
- [148] K. Panitzek, M. Ikram, M. Mühlhäuser, and T. Strufe, "Smooth Resilient Service Provision in Large Heterogeneous Networks," *PIK – Praxis der Informationsverarbeitung und Kommunikation*, vol. 35, no. 3, pp. 167–173, 2012.
- [149] K. Panitzek, M. Ikram, and M. Stein, "Service Overlays," in *Benchmarking Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, W. Effelsberg, R. Steinmetz, and T. Strufe, Eds. Springer Berlin Heidelberg, 2013, vol. 7847, pp. 113–140.
- [150] K. Panitzek, I. Schweizer, T. Bönning, G. Seipel, and M. Mühlhäuser, "Enhancing Robustness of First Responder Communication in Urban Environments," in *Proceedings of the 9th International Conference on Information Systems for Crisis Response and Management*, 2012.
- [151] —, "First Responder Communication in Urban Environments," *International Journal on Mobile Network Design and Innovation*, vol. 4, no. 2, pp. 109–118, Aug. 2012.
- [152] K. Panitzek, I. Schweizer, D. Bradler, and M. Mühlhäuser, "City Mesh – Resilient first responder communication," in *Proceedings of the 8th International Conference on Information Systems for Crisis Response and Management*, 2011.
- [153] K. Panitzek, I. Schweizer, A. Schulz, T. Bönning, G. Seipel, and M. Mühlhäuser, "Can We Use Your Router, Please? - Benefits and Implications of an Emergency Switch for Wireless Routers," *International Journal of Information Systems for Crisis Response and Management*, vol. 4, no. 4, pp. 59–70, Dec 2012.
- [154] K. Panitzek and T. Strufe, "A Formal Model for Peer-to-Peer Systems," in *Benchmarking Peer-to-Peer Systems*, ser. Lecture Notes in Computer Science, W. Effelsberg, R. Steinmetz, and T. Strufe, Eds. Springer Berlin Heidelberg, 2013, vol. 7847, pp. 15–18.
- [155] K. Panitzek, P. Weisenburger, I. Schweizer, and M. Mühlhäuser, "Towards an Integrated Mobility Simulation for Communications Research," in *Proceedings of the International Conference on Mobile Services, Resources, and Users*, 2013, pp. 24–29.

-
- [156] C. Perkins, S. Ratliff, and J. Dowdell, "Dynamic MANET On-demand (AODVv2) Routing," *Internet-Draft: draft-ietf-manet-aodvv2-03*, pp. 1–51, 2010.
- [157] C. Perkins, E. Belding-Royer, and S. Das, "RFC 3561 (Experimental) - Ad hoc On-Demand Distance Vector (AODV) Routing," *Internet RFCs*, pp. 1–37, 2003.
- [158] C. E. Perkins and P. Bhagwat, "Routing Over Multi-Hop Wireless Network of Mobile Computers," in *Mobile Computing*, ser. The Kluwer International Series in Engineering and Computer Science, T. Imielinski and H. Korth, Eds. Springer US, 1996, vol. 353, pp. 183–205.
- [159] H. Pucha, S. Das, and Y. Hu, "Ekta: An Efficient DHT Substrate for Distributed Applications in Mobile Ad Hoc Networks," in *Proceedings of the 6th IEEE Workshop on Mobile Computing Systems and Applications*, Dec 2004, pp. 163–173.
- [160] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall PTR New Jersey, 1996, vol. 2.
- [161] D. Reina, S. T. Marín, N. Bessis, F. Barrero, and E. Asimakopoulou, "An Evolutionary Computation Approach for Optimizing Connectivity in Disaster Response Scenarios," *Applied Soft Computing*, vol. 13, no. 2, pp. 833–845, Feb. 2013.
- [162] O. Riva, T. Nadeem, C. Borcea, and L. Iftode, "Context-Aware Migratory Services in Ad Hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 12, pp. 1313–1328, Dec. 2007.
- [163] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems," in *Middleware 2001*, ser. Lecture Notes in Computer Science, R. Guerraoui, Ed. Springer Berlin Heidelberg, 2001, vol. 2218, pp. 329–350. [Online]. Available: http://dx.doi.org/10.1007/3-540-45518-3_18
- [164] E. Royer, P. Melliar-Smith, and L. Moser, "An Analysis of the Optimum Node Density for Ad Hoc Mobile Networks," in *Proceedings of the IEEE International Conference on Communications*, 2001, pp. 857–861.
- [165] F. Sailhan and V. Issarny, "Scalable Service Discovery for MANET," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*. IEEE, March 2005, pp. 235–244.
- [166] R. Salami, C. Laflamme, J.-P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, and Y. Shoham, "Design and Description of CS-ACELP: A Toll Quality 8 kb/s Speech Coder," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 2, pp. 116–130, Mar. 1998.
- [167] F. Samimi, P. Mckinley, S. Sadjadi, C. Tang, J. Shapiro, and Z. Zhou, "Service Clouds: Distributed Infrastructure for Adaptive Communication Services," *IEEE Transactions on Network and Service Management*, vol. 4, no. 2, pp. 84–95, Sep. 2007.
- [168] M. Sánchez and P. Manzoni, "ANEJOS: A Java Based Simulator for Ad Hoc Networks," *Future Generation Computer Systems*, vol. 17, no. 5, pp. 573–583, Mar. 2001.

-
- [169] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.
- [170] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic Fine-grained Localization in Ad-Hoc Networks of Sensors," in *Proceedings of the 7th Annual ACM International Conference on Mobile Computing and Networking*, New York, NY, USA, Jul. 2001, pp. 166–179.
- [171] G. Schiele, C. Becker, and K. Rothermel, "Energy-efficient cluster-based service discovery for ubiquitous computing," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*. New York, NY, USA: ACM, 2004, p. 14.
- [172] B. Schiller, D. Bradler, I. Schweizer, M. Mühlhäuser, and T. Strufe, "GTNA: A Framework for the Graph-theoretic Network Analysis," in *Proceedings of the Spring Simulation Multiconference*. San Diego, CA, USA: ACM Press, Apr. 2010, pp. 111:1–111:8.
- [173] J. Schreiber, *Arbeitsanweisungen für SEGen im Sanitäts- und Rettungsdienst*. Stumpf & Kossendey, 2000.
- [174] B. Seshasayee, R. Nathuji, and K. Schwan, "Energy-Aware Mobile Service Overlays: Cooperative Dynamic Power Management in Distributed Mobile Systems," in *Proceedings of the Fourth International Conference on Autonomic Computing*. IEEE, Jun. 2007, pp. 6–6.
- [175] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, "A First Look at Cellular Network Performance During Crowded Events," *SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 17–28, Jun. 2013.
- [176] D. Sharp, N. Cackov, N. Laskovic, Q. Shao, and L. Trajkovic, "Analysis of Public Safety Traffic on Trunked Land Mobile Radio Systems," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 7, pp. 1197–1205, Sep. 2004.
- [177] A. C. Snoeren, D. G. Andersen, and H. Balakrishnan, "Fine-grained Failover Using Connection Migration," in *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, vol. 1, 2001, pp. 19–19.
- [178] A.-K. Souley and S. Cherkaoui, "Advanced Mobility Models for Ad Hoc Network Simulations," in *Proceedings of Systems Communications*. IEEE, Aug 2005, pp. 50–55.
- [179] M. R. Souryal, A. Wapf, and N. Moayeri, "Rapidly-Deployable Mesh Network Testbed," in *Proceedings of the IEEE Global Telecommunications Conference*. IEEE, Nov 2009, pp. 1–6.
- [180] P. Stavroulakis, *Terrestrial Trunked Radio - Tetra*. Springer Berlin Heidelberg, 2007.
- [181] M. Stein, "Platzierung von Softwarediensten für den Ersthelfereinsatz in mobilen Ad-hoc-Netzwerken," Telecooperation, Technische Universität Darmstadt, Bachelor's Thesis, 2012.

-
- [182] I. Stepanov, J. Hahner, C. Becker, and K. Rothermel, "A Meta-model and Framework for User Mobility in Mobile Networks," in *Proceedings of the 11th IEEE International Conference on Networks*. IEEE, Sept 2003, pp. 231–238.
- [183] D. Stingl, C. Groß, J. Rückert, L. Nobach, A. Kovacevic, and R. Steinmetz, "Peerfact-sim.kom: A simulation framework for peer-to-peer systems," in *Proceedings of the International Conference on High Performance Computing and Simulation*, July 2011, pp. 577–584.
- [184] D. Stingl, B. Richerzhagen, F. Zollner, C. Gross, and R. Steinmetz, "PeerfactSim.KOM: Take it Back to the Streets," in *Proceedings of the International Conference on High Performance Computing and Simulation*, July 2013, pp. 80–86.
- [185] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," *SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 149–160, Aug. 2001.
- [186] K. Takasugi, M. Nakamura, S. Tanaka, and M. Kubota, "Seamless Service Platform for Following a User's Movement in a Dynamic Network Environment," in *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*. IEEE Comput. Soc, March 2003, pp. 71–78.
- [187] E. Tews and M. Beck, "Practical Attacks Against WEP and WPA," in *Proceedings of the Second ACM Conference on Wireless Network Security*. New York, NY, USA: ACM, 2009, pp. 79–86.
- [188] E. Tews, R.-P. Weinmann, and A. Pyshkin, "Breaking 104 Bit WEP in Less Than 60 Seconds," in *Information Security Applications*, ser. Lecture Notes in Computer Science, S. Kim, M. Yung, and H.-W. Lee, Eds. Springer Berlin Heidelberg, 2007, vol. 4867, pp. 188–202.
- [189] T. Todd, A. Sayegh, M. Smadi, and D. Zhao, "The Need for Access Point Power Saving in Solar Powered WLAN Mesh Networks," *IEEE Network*, vol. 22, no. 3, pp. 4–10, May 2008.
- [190] P. Uppuluri, N. Jabisetti, U. Joshi, and Y. Lee, "P2P Grid: Service Oriented Framework for Distributed Resource Management," in *Proceedings of the IEEE International Conference on Services Computing*, vol. 1, 2005, pp. 347–350.
- [191] L. M. Vaquero, L. Roderio-Merino, J. Caceres, and M. Lindner, "A Break in the Clouds: Towards a Cloud Definition," *SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, Dec. 2008.
- [192] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications Networks and Systems*. ICST, Brussels, Belgium: ICST, 2008, pp. 60:1–60:10.
- [193] C. Ververidis and G. Polyzos, "Service Discovery for Mobile Ad Hoc Networks: A Survey of Issues and Techniques," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 3, pp. 30–45, 2008.

-
-
- [194] K. Wang and B. Li, "Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad-hoc Networks," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, no. c. IEEE, 2002, pp. 1089–1098.
- [195] M. Wattenhofer and R. Wattenhofer, "Distributed Weighted Matching," in *Distributed Computing*, ser. Lecture Notes in Computer Science, R. Guerraoui, Ed. Springer Berlin Heidelberg, 2004, vol. 3274, pp. 335–348.
- [196] E. Weingartner, H. vom Lehn, and K. Wehrle, "A Performance Comparison of Recent Network Simulators," in *Proceedings of the IEEE International Conference on Communications*, June 2009, pp. 1–5.
- [197] P. Weisenburger, "Diskret ereignisreiche, rollenbasierte Bewegungssimulation in Disaster Scenarios," Telecooperation, Technische Universität Darmstadt, Bachelor's Thesis, 2011.
- [198] G. Wittenburg, "Service Placement in Ad Hoc Networks," Ph.D. dissertation, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 2010.
- [199] G. Wittenburg and J. Schiller, "A Survey of Current Directions in Service Placement in Mobile Ad-hoc Networks," in *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications*. IEEE, Mar. 2008, pp. 548–553.
- [200] F. Wolf, J. Balasubramanian, S. Tambe, A. Gokhale, and D. C. Schmidt, "Supporting Component-based Failover Units in Middleware for Distributed Real-time and Embedded Systems," *Journal of Systems Architecture*, vol. 57, no. 6, pp. 597 – 613, 2011.
- [201] H. Xu and B. Li, "Egalitarian Stable Matching for VM Migration in Cloud Computing," in *Proceedings of the IEEE Conference on Computer Communications Workshops*. IEEE, April 2011, pp. 631–636.
- [202] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," *SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 379–390, Aug. 2004.
- [203] J. Yoon, M. Liu, and B. Noble, "Random Waypoint Considered Harmful," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, March 2003, pp. 1312–1321.
- [204] T. Zahn and J. Schiller, "MADPastry: A DHT Substrate for Practicably Sized MANETs," in *Proceedings of ASWN*, 2005.
- [205] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks," in *Proceedings of the Twelfth Workshop on Parallel and Distributed Simulation*. IEEE Comput. Soc, May 1998, pp. 154–161.
- [206] Y. Zhou, J. Cao, V. Raychoudhury, J. Siebert, and J. Lu, "A Middleware Support for Agent-Based Application Mobility in Pervasive Environments," in *Proceedings of the 27th International Conference on Distributed Computing Systems Workshops*. IEEE, June 2007, pp. 9–9.

-
- [207] S. Zhu, W. Wang, and C. Ravishankar, “A New Power-Efficient Scheme to Deliver Time-Sensitive Data in Sensor Networks,” in *Proceedings of the IEEE International Conference on Mobile Adhoc and Sensor Systems*, Oct 2006, pp. 188–198.



Curriculum Vitae

PERSONAL INFORMATION	Kamill Christoph Panitzek born August 23, 1983 in Strzelce, Poland
2010 – 2014	Doctorate <i>Telecooperation Lab, Computer Science</i> <i>Technische Universität Darmstadt, Germany</i> Research Area: Smart Urban Networks Title: Mobile Service Provision in Harsh Environments 1 st Referee: Prof. Dr. Max Mühlhäuser (Technische Universität Darmstadt, Germany) 2 nd Referee: Prof. Dr. Jussi Kangasharju (University of Helsinki, Finland)
2008 – 2010	Master of Science in computer science <i>Technische Universität Darmstadt, Germany</i> Minor: Biology Thesis Title: Monitoring File Popularity in Peer-to-Peer Networks Advisors: Prof. Dr. Max Mühlhäuser Dirk Bradler (Technische Universität Darmstadt, Germany)
2004 – 2008	Bachelor of Science in computer science <i>Technische Universität Darmstadt, Germany</i> Thesis Title: Movement Models in a Development Environment for Disaster Scenarios Advisors: Prof. Dr. Max Mühlhäuser Dirk Bradler (Technische Universität Darmstadt, Germany)
2003 – 2004	Civilian service <i>Institut für Arbeitswissenschaften, Mechanical Engineering,</i> <i>Technische Universität Darmstadt, Germany</i>
1994 – 2003	German diploma for university admission (Allgemeine Hochschulreife) <i>Lichtenbergschule, Darmstadt, Germany</i>